

ENKCH-1.0

VAX 11/730
IDC SIZER

EP-ENKCH-DL-1.0
FICHE 1 OF 1

JUL 1983
COPYRIGHT © 1983
MADE IN USA



IDENTIFICATION

Product code: ZZ-ENKCH VERSION 1.0
Product name: VAX-11/730 CRD IDC MICRO-SIZER
Product date: 27-AUG-1982
Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS

DECsystem-10
MASSBUS
VAX

DECSYSTEM-20
PDP
VMS

digital

Table of Contents

1.0	ABSTRACT	3
2.0	MAINTENANCE HISTORY	3

1.0 ABSTRACT

This program is a microdiagnostic that is used during VAX-11/730 Customer Runnable Diagnostics AUTO mode. It is to be run under the CRD Micmon only, and is not to be used under apt or as a debugging tool. This program's function is to test if an IDC is present; to flag CRD MICMON if it should run the IDC microdiagnostics ENKCF and ENKCG in CRD AUTO MODE.

For further information about the program, and for complete documentation concerning CRD AUTO mode, consult the ENSAB.DOC documentation file.

2.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
23-MAY-83	01.00	Initial release to support LCN and KERNEL 11/730 systems.

715	FIELD DEFINITIONS FOR MAIN MICRO WORD	VER 00.08
1721	MACRO DEFINITIONS	VER 00.02
2377	FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD	VER 00.01
2430	CONTROL ROM CONTENTS	VER 00.01
3179	DIAGNOSTIC MACROS AND DEFINITIONS	
3838	COMMON LS ASSIGNMENTS (TO REGISTERS)	
3998	Microinstruction Formats	
4381	Tables	
4512	2901 ALU Control Description	
4616	DIAGNOSTIC MACROS	
4644	TEST POINTERS	
4694	DIAGNOSTIC POINTERS	
4784	LS DATA SECTION	
5175	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	
5547	WCS SUBROUTINES FOR CONSOLE USE	
5548	GENERATE TRANSFER DATA	
5594	DEPOSIT MCT CSR ROUTINE	
5660	EXAMINE MCT CSR ROUTINE	
5747	DEPOSIT MCT TRANSLATION BUFFER ROUTINE	
5863	EXAMINE TRANSLATION BUFFER ROUTINE	
5926	DEPOSIT UNIBUS MAP ROUTINE	
6024	EXAMINE UNIBUS MAP ROUTINE	
6087	DEPOSIT MAIN MEMORY ROUTINE	
6145	EXAMINE MAIN MEMORY ROUTINE	
6207	EXAMINE IDC ROUTINES	
6273	DEPOSIT IDC ROUTINES	
6345	SAVE WR ROUTINE	
6396	RESTORE WR ROUTINE	
6447	WCS SUBROUTINES FOR CPU USE	
6448	ERROR ROUTINE	
6615	START TRANSFER ROUTINE	
6713	Test 1 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)	

```

:1  TITLE "ENKCH Micro-diagnostic for IDC module Rev 01.00"
:2
:3
:4  COPYRIGHT (c) 1982 BY
:5  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
:6  MASSACHUSETTS. ALL RIGHTS RESERVED.
:7
:8  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
:9  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
:10 OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER COPIES THEREOF
:11 MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON. NO
:12 TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
:13
:14 THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
:15 SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
:16
:17 DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
:18 SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
:19
:20
:21 ++
:22 FACILITY: VAX-11/730 MICRO-DIAGNOSTIC.
:23
:24 ABSTRACT: This micro-diagnostic tests the hardware of the IDC module
:25 in the VAX-11/730 processor.
:26
:27 ENVIRONMENT: ENKAA Micro-diagnostic Monitor
:28
:29 AUTHOR: PETER GREEN 24-MARCH-83 VERSION 1.0
:30
:31 Initial release.
:32
:33 MODIFIED BY:
:34
:35 --
:36 .nolist
:40 .list
:41 .NOCREF
  
```

```

:42 .NOBIN
:43 .SEQUENTIAL
:44 .RTOL
:45 .HEXADECIMAL
:46 REVISION HISTORY
:47
:48 VER 02.24 9-SEP-80 SCS
:49
:50 ADD THE BIC WR[] WITH WR[] TO Q MACRO.
:51
:52 VER 02.23 16-JUL-80 SCS
:53
:54 FIX MINC INSTRUCTION TO CALL INC INSTEAD OF DEC
:55
  
```



```

:56 : VER 02.22 15-JUL-80 SCS
:57 :
:58 :   ADD SAVED.MDT TO LOCAL STORE.
:59 :
:60 : VER 02.21 7-JUL-80 SCS
:61 :
:62 :   REMAP ALL OF THE CONSTANTS TO THE LOCAL STORE LOCATIONS
:63 :   REQUIRED BY THE 8085 TO BUILD THEM DURING POWER-UP.
:64 :
:65 :   THIS ALLOWS US TO RECOVER FROM POWER FAIL CORRECTLY.
:66 :
:67 :   THE CHANGES BASICALLY REQUIRED A RESHUFFLING OF THE CONSTANTS
:68 :   ACCORDING TO SIZE OF THE NUMBER.
:69 :
:70 : VER 02.20 13-JUN-80 SCS
:71 :
:72 :   FIX VALIDITY CHECKS TO ALLOW LEGAL SKIP CONDITIONS ON
:73 :   THE PAGE BOUNDARY CONSISTING OF:
:74 :
:75 :   RETURN
:76 :   RETURN+1
:77 :   RET.NOERR
:78 :   POP.USTACK
:79 :
:80 : VER 02.19 04-JUN-80 SCS
:81 :
:82 :   ADD VALIDITY CONDITIONS TO THE SKIP AND JUMP CONTROL
:83 :   FIELDS TO CHECK FOR THE PAGE PROBLEM CREATED BY THE SEQUENCER. THIS
:84 :   LIMITATION ONLY ALLOWS CARRIES TO BE PROPAGATED FOR 10 BITS WHEN
:85 :   THE ADDRESS IS INCREMENTED BY TWO. THE NORMAL SEQUENCING OF INSTRUCTIONS
:86 :   CAN WALK ACROSS THESE 1K BOUNDARIES.
:87 :
:88 :   THIS REQUIRES ADDITION OF THE FOLLOWING VALIDITY CHECKS AND CHANGES
:89 :   TO THE SCTL AND JCTL FIELD'S VALIDITY CHECKS.
:90 :
:91 :   PAGE.PROB
:92 :   SKIP.PAGE.VAL
:93 :   JUMP.CHECK
:94 :   JUMP.PAGE.VAL
:95 :
:96 : VER 02.18 02-JUN-80 SCS
:97 :
:98 :   REMOVE THE MACRO BIC WR[] WITH LS[] TO 0
:99 :   SINCE IT IS NOT IMPLEMENTED ANYMORE.
:100 :
:101 : VER 02.17 13-MAY-80 SCS
:102 :
:103 :   ADD BIT TO LS[CONSOLE.CSR.IES]
:104 :
:105 :   <5> - CRD RETRY IN PROGRESS.
:106 :
:107 :   ADD LS LOCATION CRD.LOG
:108 :
:109 : VER 02.16 18-APR-80 SCS
:110 :
  
```

:111 : CHANGE MEM.FUNC FIELD TO CONTAIN:
:112 :
:113 : READ.MAINT.BRANCH.CHECK
:114 : READ.MAINT.UBS
:115 :
:116 : UPDATE MOV FPA TO LS[] TO REMOVE CC SETTING.
:117 :
:118 : VER 02.15 14-APR-80 SCS
:119 :
:120 : ADD THE FOLLOWING FIELD VALUES TO THE MEM.FUNC FIELD:
:121 :
:122 : READ.MAINT.VAR.INC
:123 : OCTA.WRITE.P
:124 : MAINT.ECC.DATA
:125 : OCTA.READ.P
:126 :
:127 : CHANGE THE DATA PATH CONTROL ROM TO MATCH THE PRESENT HARDWARE
:128 : IMPLEMENTATION. REMOVE THE PARWORD AND PAR2 FIELDS. CHANGE
:129 : FIELD VALUES FOR THE SRC FIELD.
:130 :
:131 : CHANGE SRC FIELDS OF THE COND.ADD&SHIFT AND COND.SUB&SHIFT MACROS
:132 : TO USE A.B AND NOT THE NEW DEFAULT MUL FIELD VALUE. THE HARDWARE
:133 : NOW OVERRIDES WHATEVER IS IN THE FIELD.
:134 :
:135 : VER 02.14 19-MAR-80 SCS
:136 :
:137 : ADD SUPPORT FOR THE IDC.!!.
:138 :
:139 : THE FOLLOWING FIELDS HAVE BEEN ADDED WITH THEIR ASSOCIATED
:140 : VALUES:
:141 :
:142 : PORT.SELECT
:143 : PORT.FUNC
:144 : R.W.FUNC
:145 : CNTL.FUNC
:146 :
:147 : THE FOLLOWING MACROS HAVE BEEN ADDED:
:148 :
:149 : PORT.CONTROL
:150 : PORT.WRITE PORT.ADRS[] WITH WR[]
:151 : PORT.READ PORT.ADRS[]
:152 :
:153 : VER 02.13 13-MAR-80 SCS
:154 :
:155 : ADD THE FOLLOWING CONSTANTS TO THE LOCAL STORE:
:156 :
:157 : #40A10
:158 :
:159 : ADD LOCATION IN LOCAL STORE:
:160 :
:161 : MEMORY.SIZE
:162 :
:163 : VER 02.12 23-FEB-80 SCS
:164 :
:165 : ADD THE FOLLOWING NEW LABELS TO OLD LOCAL STORE

```
:166 : LOCATIONS:
:167 :
:168 : CONSOLE.COMMAND
:169 : CONSOLE.MODIFIER
:170 : CONSOLE.ADDRESS
:171 : CONSOLE.STATUS
:172 : CONSOLE.DATA
:173 :
:174 : VER 02.11 11-FEB-80 SCS
:175 :
:176 : UPDATE COMMENTS TO SWAP AND MEM FUNCTION.
:177 :
:178 : VER 02.10 7-FEB-80 SCS
:179 :
:180 : CHANGED CM.EXEC ADRS[] TO LOAD THE OS REGISTER.
:181 :
:182 : VER 02.09 17-DEC-79 SCS
:183 :
:184 : ADD THE CROSS REFERENCE LISTING TO OUTPUT.
:185 :
:186 : BUILD THE FOLLOWING MEMORY CODES FOR THE MEM.FUNC FIELD:
:187 :
:188 : WRITE.TB.STEP=11
:189 : OCTA.READ.V.RCHK=1A
:190 : READ.MAINT.UBS.DATA=1B
:191 : OCTA.WR.V.WCHK=1D
:192 : QUAD.READ.V.RCHK=1E
:193 :
:194 : VER 02.08 12-NOV-79 SCS
:195 :
:196 : CREATE BACKUP LOCATION FOR COMPATIBILITY MODE PC.
:197 :
:198 : BACKUP.CM.PC
:199 :
:200 : ADD THE FOLLOWING CONSTANTS:
:201 :
:202 : #FF80
:203 : #FFF0
:204 : #40211
:205 : #66
:206 :
:207 : VER 02.07 30-OCT-79 SCS
:208 :
:209 : CHANGE LS LOCATIONS
:210 :
:211 : #2D40
:212 : #C00000E0
:213 : #FFFFFF1F
:214 : TO
:215 : #2D20
:216 : #C00000E0
:217 : #FFFFFF10
:218 :
:219 : VER 02.06 29-OCT-79 SCS
:220 :
```


:221 : CHANGE LS LOCATION
:222 :
:223 : TO #FFFFFF8F
:224 :
:225 : #FFFFFF1F
:226 :
:227 : VER 02.05 26-OCT-79 SCS
:228 :
:229 : CHANGE LS LOCATIONS
:230 : #5F
:231 : #5F40
:232 :
:233 : TO
:234 : #2D
:235 : #2D40
:236 :
:237 : VER 02.04 26-OCT-79 SCS
:238 :
:239 : FIX THE FOLLOWING MACROS CALL TO DATA PATH CONTROL ROM.
:240 :
:241 : ADD Q TO WR[] XCHG TO LS[]
:242 :
:243 : VER 02.03 11-OCT-79 SCS
:244 :
:245 : ADD
:246 : SAVED.2ND.REQUEST
:247 : PSEUDO.NICR
:248 : PSEUDO.ICR
:249 : TO LS LOCATIONS.
:250 :
:251 : FIX ERROR IN
:252 : MOV FPA TO LS[].
:253 :
:254 : VER 02.02 04-OCT-79 SCS
:255 :
:256 : FIX DEFAULT FOR BIT <18> TO BE 1 FOR DECODE INSTRUCTIONS AND
:257 : 0 FOR ALL OTHERS.
:258 :
:259 : VER 02.01 27-SEP-79 SCS
:260 :
:261 : FIX THE BPC FIELD DEFAULT.
:262 :
:263 : VER 02.00 13-SEP-79 SCS
:264 :
:265 : THIS IS THE VERSION WHICH PROVIDES COMPATIBILITY BETWEEN
:266 : BREADBOARD ONE AND BREADBOARD TWO.
:267 :
:268 : VALUES OF THE BIC FIELD ARE COMPLEMENTED.
:269 :
:270 : MISC.2 FIELD VALUES ARE RE-ARRANGED.
:271 :
:272 : MISC.4 FIELD REMOVED.
:273 :
:274 : CHANGES TO THE FOLLOWING MACROS:
:275 :

```

:276 :      MOV WR[] TO WR[]
:277 :      MOV WR[] TO Q
:278 :      CLR Q
:279 :      CLR WR[]
:280 :      NEG WR[]
:281 :      INC WR[]
:282 :      DEC WR[]
:283 :      COM WR[]
:284 :      TST WR[]
:285 :
:286 :      ADD THE FOLLOWING MACROS
:287 :
:288 :      MCOM WR[] TO WR[]
:289 :      MINC WR[] TO WR[]
:290 :      MDEC WR[] TO WR[]
:291 :
:292 :      VER 01.23      11-SEP-79      SCS
:293 :
:294 :      ADD THE FOLLOWING CONSTANTS
:295 :
:296 :      #F0
:297 :      #A0
:298 :      #30
:299 :      #5F
:300 :      #5F40
:301 :
:302 :      UPDATE TO FIELD VALUE:
:303 :
:304 :      ASSERT.DA.AND.PASS.WR
:305 :
:306 :      RENAME NEW FIELDS:
:307 :
:308 :      MISC.WRL
:309 :      MISC.WRR
:310 :
:311 :      ADD NEW MACROS:
:312 :
:313 :      MOV FPA TO LS[]
:314 :      MOV PORT TO LS[]
:315 :      ASSERT.DA.AND.PASS.WR0
:316 :      ASSERT.DA.AND.PASS.WR1
:317 :      ADD Q TO WR[] XCHG TO LS[]
:318 :
:319 :      CHANGE MACROS:
:320 :
:321 :      MISC[] WR[]
:322 :      MISC2[] WR[]
:323 :
:324 :      REMOVE MACROS:
:325 :      (ALL PRIORITY ENCODE FUNCTIONS)
:326 :
:327 :      RENAME MACRO TO
:328 :
:329 :      LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
:330 :

```

ZZ-ENKCH-1.0 ;
ENKCH.MCR
ENKCH.MIC

ENKCH.MIC
MICRO2 1M(01)

19-MAY-83 13:28:52

19-MAY-83

ENKCH Micro-diagnostic for IDC module

Fiche 1 Frame L1

Sequence 11
Rev 01.00

Page 8

```
331 : VER 01.22 02-AUG-79 SCS
332 :
333 :     ADD HARDWARE INDEXES TO PORT REGISTERS.
334 :
335 : VER 01.21 31-JUL-79 SCS
336 :
337 :     CHANGED CONSTANT #F03000 TO #F27000
338 :     ADDED NEW MACROS
339 :         MISC[] WR[]
340 :         MISC2[] WR[]
341 :
342 : VER 01.20 20-JUL-79 SCS
343 :
344 :     ADD FUNCTION
345 :
346 :         MISC2[MASK.INTERRUPTS]
347 :
348 : VER 01.19 16-JUL-79 SCS
349 :
350 :     ADD NEW MACRO
351 :
352 :         MOV CM.IB.DATA TO OS
353 :
354 :     ADD TRUE LSS TO SCTL FIELD.
355 :     RENAME FALSE LSS TO N.CLR
356 :         GEQ TO N.SET
357 :     IN BOTH THE SCTL FIELD AND JCTL FIELD.
358 :     THERE ARE NOW NO FREE SKIP CONDITIONS!!
359 :
360 : VER 01.18 13-JUL-79 SCS
361 :
362 :     FIX ERROR WITH VALIDITY ON XCHG.BIT
363 :
364 : VER 01.17 07-JUL-79 SCS
365 :
366 :     CHANGED #3A3000 TO #F03000
367 :     INSERT ROTATE.BYTE.RIGHT INTO MEM.FUNC FIELD.
368 :
369 :     ADD SYMMETRIC MACROS:
370 :
371 :         SWAP WR[] WITH LS[]
372 :         ADD LS[] PLUS WR[] TO Q
373 :         BIS LS[] WITH WR[] TO Q
374 :         AND LS[] WITH WR[] TO Q
375 :         XOR LS[] WITH WR[] TO Q
376 :
377 :     ADD MNEG Q TO LS[]
378 :
379 : VER 01.16 08-JUN-79 SCS
380 :
381 :     DEFINE THE FOLLOWING LOCATIONS:
382 :
383 :         IE.TEMP1
384 :         IE.TEMP2
385 :         IE.TEMP3
```



```

:386      IE.TEMP4
:387
:388      #B020FF00
:389      #C00000E0
:390
:391      RENAME SKIP CONDITION IN SCTL FIELD TO:
:392
:393      NO.FAST.INTERRUPT
:394
:395      VER 01.15      16-MAY-79      SCS
:396
:397      ADD LD.OS/LOAD.OS TO
:398      DECODE.CM.OPC ADRS[]
:399
:400      VER 01.14      14-MAY-79      SCS
:401
:402      ADD A ZERO LOCATION TO LOCAL STORE.
:403
:404      VER 01.13      08-MAY-79      SCS
:405
:406      DEFINE TWO NEW FUNCTIONS FOR SI FIELD.
:407
:408      UMUL.SHF
:409      SHF.WR.Q
:410
:411      ADD THE FOLLOWING MACROS:
:412
:413      UNSIGNED.MUL WR[] TO WR[].Q
:414      SHR WR[]
:415      SHL WR[]
:416      SHRC WR[].Q
:417      SHLC WR[].Q
:418
:419      VER 01.12      01-MAY-79      SCS
:420
:421      INSERT LS[CONSOLE.CSR.IES]
:422
:423      VER 01.11      19-APR-79      SCS
:424
:425      ADDITION OF EQL PNEUMONIC TO THE JCTL FIELD. CHANGE
:426      TO DPCROM TO FIX THE FOLLOWING MACROS:
:427
:428      ADD Q PLUS LS[] TO WR[]
:429      COND.ADD&SHIFT WR[] TO WR[].Q
:430
:431      VER 01.10      10-APR-79      SCS
:432
:433      ADDITION OF .SET/UPC=000 TO THE DEFINITION FILE SO THAT ALL
:434      MICRO ASSEMBLERS CAN USE .CHANGE/UPC=<.+1> IN THEIR CODE.
:435
:436      VER 01.09      04-APR-79      SCS
:437
:438      FIX DECODE.CM.DST ADRS[]
:439
:440      ADD SAVE.CM.PC WRO

```

```

:441 :
:442 :      UPDATE MOV IB.DATA TO OS
:443 :
:444 :      SWITCH JCTL FIELDS
:445 :          NO.JUMP.TST
:446 :          JUMP.VALID
:447 :
:448 :      VER 01.08      30-MAR-79      SCS
:449 :
:450 :      ADD IN OS.BAK
:451 :
:452 :      ADD IN TST Q
:453 :
:454 :      VER 01.07      21-MAR-79      SCS
:455 :
:456 :      REMOVE THE FOLLOWING IMPOSSIBLE MACROS:
:457 :
:458 :      BIC Q TO LSC[]
:459 :      BIC WR[] TO LSC[]
:460 :      BIC Q TO WR[]
:461 :      BIC WR[] WITH LSC[] TO Q
:462 :
:463 :      UPDATE THE FOLLOWING MACROS TO WORK:
:464 :
:465 :      OPR Q TO WR[]
:466 :
:467 :      ADD THE FOLLOWING MACRO:
:468 :
:469 :      ADD Q PLUS LSC[] TO WR[]
:470 :
:471 :      VER 01.06      19-MAR-79      SCS
:472 :
:473 :      ADD NEW LOCATIONS TO LOCAL STORE FOR MEMORY MANAGEMENT
:474 :      ADD TWO NEW TEMPORARY LOCATIONS.
:475 :      FIX UP DPCROM FOR
:476 :
:477 :      PRI.ENC WR0 R.TO.L TO OS AND CLEAR BIT
:478 :
:479 :      ADD THE FOLLOWING CONSTANTS:
:480 :
:481 :      #FF
:482 :      #FFFFFFFF
:483 :      #3A3000
:484 :      #7FFFFFFE00
:485 :      #3FFFFFFE00
:486 :      #38
:487 :      #3F
:488 :      #3000
:489 :      #FFFFFF000
:490 :      #FFE0
:491 :      #FFF
:492 :      #0FFF0000
:493 :      #FFFFFF8F
:494 :
:495 :      ADD THE FOLLOWING MACROS:
  
```

```
:496 :  
:497 : ADD Q PLUS WR[] TO LS[]  
:498 : SUB Q FROM WR[] TO LS[]  
:499 :  
:500 : FIX UP ERROR IN MACRO:  
:501 :  
:502 : MOV XWR[] TO Q  
:503 : CMP Q WITH WR[]  
:504 :  
:505 : VER 01.05 13-MAR-79 SCS  
:506 :  
:507 : FIX MEM.FUNC FIELD TO MATCH HARDWARE.  
:508 :  
:509 : FIX ERROR IN PRI.ENC MACRO IN DATA PATH CONTROL ROM.  
:510 :  
:511 : FIX ERROR IN MISC.2 FIELD  
:512 :  
:513 : VER 01.04 07-MAR-79 SCS  
:514 :  
:515 : FIX ERROR IN THE LABEL IN DATA PATCH CONTROL ROM  
:516 : CORRESPONDING TO THE FOLLOWING MACRO:  
:517 :  
:518 : BIS WR[] WITH LS[] TO Q  
:519 :  
:520 : CHANGE TST WR[] MACRO TO USE NEW DATA PATH CONTROL ROM FUNCTION  
:521 : WHICH WILL CLEAR THE V AND C CONDITION CODES IF DT(SIZE)&SET.ALU.CC  
:522 : IS INVOKED.  
:523 :  
:524 : INSERT NEW MACROS FOR COMPATIBILITY MODE DECODE MACROS:  
:525 :  
:526 : JMP.TO []  
:527 : JSR.TO []  
:528 :  
:529 : FIX ERROR IN:  
:530 :  
:531 : DECODE.CM.SINGLE ADRS[]  
:532 :  
:533 : VER 01.03 05-MAR-79 SCS  
:534 :  
:535 : FIX ERROR IN THE FOLOWING MACROS:  
:536 :  
:537 : SET.STATE.ZERO&CLR.STATE.ONE  
:538 : SET.STATE.ONE&CLR.STATE.ZERO  
:539 :  
:540 : INSERT CM.PC LOCATION OF THE LOCAL STORE.  
:541 :  
:542 : INSERT THE FOLLOWING MACROS:  
:543 :  
:544 : LOOP.IF(NO INTERRUPT AND NO SYNC)  
:545 :  
:546 : VER 01.02 01-MAR-79 SCS  
:547 :  
:548 : IMPLEMENT NEW PRIORITY ENCODE FUNCTIONS AND DOCUMENT:  
:549 :  
:550 : PRI.ENC WRO R.TO.L TO OS
```



```

:551 : PRI.ENC WRO L.TO.R TO OS
:552 : PRI.ENC WRO R.TO.L TO OS AND CLEAR BIT
:553 : PRI.ENC WRO L.TO.R TO OS AND CLEAR BIT
:554 :
:555 : COMPLEMENT ALL FIELD VALUES OF THE MISC.1 FIELD.
:556 : ADD VAR FIELD VALUE FOR WR[5].
:557 : PUT THE FOLLOWING LOCATIONS INTO THE HIGH HALF OF LS:
:558 :     POBR
:559 :     POLR
:560 :     P1BR
:561 :     P1LR
:562 :     SBR
:563 :     SLR
:564 :
:565 : RESERVE LOCATIONS A0-BF FOR THE PORT. (PORT0-PORT31)
:566 : REMOVE TEMPORARY LOCATIONS T22-T28 LEAVING T0-T15 AS
:567 : THE TEMPORARY LOCATIONS AVAILABLE.
:568 : INSERT .PAGE 'S FOR READABILITY.
:569 :
:570 : VER 01.01      01-MAR-79      SCS
:571 :
:572 :     FIX THE SUB LS[] FROM WR[] TO Q MACRO.
:573 :
:574 : VER 01.00      23-FEB-79      SCS
:575 :
:576 : THIS VERSION REPRESENTS THE CHANGES TO MATCH THE HARDWARE CHANGES
:577 : TO BE IMPLEMENTED BEFORE MARCH 1,1979.
:578 :
:579 : DP.SMALL AND XCHG.BIT FIELDS CREATED TO HELP EXCHANGE FUNCTION.
:580 : SCTL VALUES CHANGED.
:581 : JCTL VALUES CHANGED.
:582 : IFUNC VALUES CHANGED AND CREATION OF OPC.SPEC FIELD.
:583 : MISC.PORT FIELD CREATED.
:584 : MISC.1 AND MISC.2 VALUES CHANGED.
:585 : MISC.4 VALUES CREATED.
:586 : MEM.FUNC VALUES CHANGED.
:587 : CREATION OF MISC2[] MACRO AND UPDATE OF ALL MISCELLANEOUS MACROS.
:588 : RESHUFFLING OF CONTROL ROM TO MATCH HARDWARE CHANGES AND
:589 : ADDITION OF ENTRIES TO CONTROL ROM FOR NEW MACROS.
:590 :
:591 : ADDITIONAL COMMENTS TO EXPLAIN OPERATION OF ALU CC'S FOR
:592 : ALL OPERATIONS.
:593 :
:594 : ADDITION OF THE FOLLOWING MACROS:
:595 :
:596 :     XCHG
:597 :     SWAP LS[] WITH WR[]
:598 :     MOV MEM.DATA TO WR[] XCHG TO LS[]
:599 :     SET.BACKUP.MASK.VALID
:600 :     PRIORITY ENCODE WRO R.TO.L TO OS
:601 :     PRIORITY ENCODE WRO L.TO.R TO OS
:602 :
:603 : CONVERT ALL Q.WR[] MACROS TO WR[].Q
:604 : ADD TIMER ENTRY TO LOCAL STORE.
:605 :
  
```

:606 : VER 00.17 13-FEB-79 SCS
:607 :
:608 : SWITCH READ CHECK AND WRITE CHECK FUNCTIONS FOR THE MEMORY.
:609 :
:610 : VER 00.16 12-FEB-79 SCS
:611 :
:612 : CHANGED MDT VALUES TO CORRECT ONES.
:613 : CHANGE MEMORY FUNCTION VALUES TO MATCH HARDWARE.
:614 :
:615 : VER 00.15 31-JAN-79 SCS
:616 :
:617 : SPLIT SOURCE OF DEF.MIC INTO THREE FILES:
:618 : DEFPREFIX.MIC
:619 : DEFDEF.MIC
:620 : DPCROM.MIC
:621 :
:622 : ADD CORRECT VALUES AND NAMES OF ALL MEMORY FUNCTIONS.
:623 :
:624 : VER 00.14 29-JAN-79 SCS
:625 :
:626 : UPDATE COMMENTS ABOUT PSL STATE.
:627 :
:628 : VER 00.13 15-JAN-79 SCS
:629 :
:630 : ADD MACROS FOR RORC Q.WR[] AND ROLC Q.WR[]
:631 :
:632 : VER 00.12 12-JAN-79 SCS
:633 :
:634 : FIX SHL2, ASHRC, ASHLC
:635 : DEFINED DEBUGGER LOCAL STORE LOCATIONS
:636 : DEFINED ALL MEMORY FUNCTION NAMES (VALUES NOT FINAL)
:637 :
:638 :
:639 : VER 00.00
:640 :
:641 : INITIAL RELEASE TO LIBRARY
:642 :
:643 : VER 00.01
:644 :
:645 : REDEFINE OPCODE BITS PER HARDWARE IMPLEMENTATION
:646 :
:647 : VER 00.02 19-SEPT-78
:648 :
:649 : ADD MACROS FOR STATE REGISTER CONTROL
:650 : UPDATE SKIP CONTROL FIELD FOR NEW ITERATION CONTROL FUNCTIONS
:651 : ADD MISC FUNCTIONS FOR MASKING TP AND CONSOLE HALT
:652 : ADD MISC FUNCTION FOR MEMORY RESET
:653 : ASSIGN LOCAL STORE ADDRESS FOR SOFT PSL
:654 : ASSIGN MEMORY MANAGEMENT REGISTERS
:655 : ASSIGN EXTENDED TEMP LOCATIONS FOR STACK POINTERS CONTROL BLOCK BASE
:656 : ADD NEW CONSTANTS
:657 :
:658 : VER 00.03 22-SEPT-78
:659 :
:660 : ADD EXTENDED B ADDRESS MACRO

```

:661 : ADD STATUS REGISTER NAMES
:662 : ADD ALTERNATE NAMES FOR CONSTANTS
:663 : FIX MISC [ ] MACRO
:664 : ADD CONSTANT 1F
:665 : ADD ABILITY TO SKIP ON BOTH POLARITIES OF THE STATE REGISTER
:666 :
:667 : VER 00.04 4-OCT-78
:668 :
:669 : UPDATE 2901 CONTROL PER HARDWARE IMPLSPEC
:670 : ADD CS PARITY GENERATION
:671 : CHANGE ADDRESS OF MACRO PC IN LS
:672 : UPDATE SKIP AND JUMP CONTROL ASSIGNMENTS
:673 : ADD CONSTANTS
:674 : FIX XWR MOVES
:675 :
:676 : VER 00.05 11-OCT-78
:677 :
:678 : REVISE SYNTAX FROM DESIGN REVIEW
:679 :
:680 : VER 00.06 13-OCT-78
:681 :
:682 : FIX MEM.REQ MACRO
:683 : ADD XD CONSTANTS
:684 :
:685 : VER 00.07 16-OCT-78
:686 :
:687 : FIX SCTL AND JCTL CODES FOR Z AND C BITS
:688 :
:689 : VER 00.08 20-OCT-78
:690 :
:691 : ADD MEMORY FUNCTION FOR READ CONTROL REGISTER
:692 : FIX LOOP CONTROL MACRO FOR GEQU
:693 : ADD BACKUP GPR ADDRESSES
:694 : ADD CONSTANTS
:695 :
:696 : VER 00.09 27-OCT-78
:697 :
:698 : UPDATE CONTROL FIELDS TO REFLECT NEW HARDWARE SPEC
:699 : ADD SHIFT FIELD CODES
:700 :
:701 : VER 00.10 20-DEC-78
:702 :
:703 : Add shift and rotate macros
:704 : Change all CLR macros to 'R.AND.S' from R.XNOR.S
:705 :
:706 : VER 00.11 4-JAN-79
:707 :
:708 : Eliminate redundant macros for shift & rotate
:709 : Add MNEG WR[ ] WR[ ]
:710 : Fill in gaps between groups of macros for ROM sake
:711 :
:712 :
:713 :

```



```

:714 .PAGE
:715 .TOC "FIELD DEFINITIONS FOR MAIN MICRO WORD VER 00.08"
:716 .SET/UPC=<000>
:717
:718 ; THE FOLLOWING EXPRESSIONS ARE VALIDITY CHECKS FOR FIELD COMBINATIONS
:719
:720 .SET/A.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0]>
:721 .SET/B.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:722 .SET/D.VAL=<.CASE[<OPC2/>]OF[0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1]>
:723 .SET/XD.VAL=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:724 .SET/CC.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
:725 .SET/DT.VAL=<.EQL[<OPC2/>,<OPC2/MEM.REQ>]>
:726 .SET/SCTL.VAL=<.CASE[<OPC2/>]OF[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1]>
:727 .SET/JCTL.VAL=<.CASE[<OPC2/>]OF[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]>
:728 .SET/JUMP.VAL=<.EQL[<OPC2/>,<OPC2/JUMP>]>
:729 .SET/DECODE.VAL=<.EQL[<OPC2/>,<OPC2/DECODE>]>
:730 .SET/MISC.VAL=<.EQL[<OPC2/>,<OPC2/MISC>]>
:731 .SET/DP.VAL=<.EQL[<OPC/>,<OPC/BASIC>]>
:732
:733 THE FOLLOWING VALIDITY CHECKS ARE USE FOR THE PAGE BOUNDARY PROBLEM.
:734
:735 .SET/PAGE.PROB=<.EQL[<.AND[<7FF>,<.]>],<7FE>]>
:736 .SET/SKIP.LEGAL2=<.OR[<.EQL[<SCTL/>,<SCTL/RET.NOERR>]>,<.EQL[<SCTL/>,<SCTL/POP.USTACK>]>]>
:737
:738 .SET/SKIP.LEGAL=<.OR[<.EQL[<SCTL/>,<SCTL/NO.SKIP>]>,<.EQL[<SCTL/>,<SCTL/RETURN>]>,<.EQL[<SCTL/>,<SCTL/RETURN+1>]>,<SKIP.LEGAL2>]>
:739
:740 .SET/SKIP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<SKIP.LEGAL>]>
:741 .SET/JUMP.CHECK=<.OR[<.EQL[<JCTL/>,<JCTL/JSR>]>,<.EQL[<JCTL/>,<JCTL/JSR.VALID>]>]>
:742
:743 .SET/JUMP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<.XOR[1,<JUMP.CHECK>]>]>
:744
:745
:746 ; MICRO INSTRUCTION OPCODE DEFINITIONS
:747
:748 OPC/=<22>,.DEFAULT=<OPC/BASIC> ; MICRO OPCODE FIELD FOR SINGLE BIT OPCODE.
:749
:750 BASIC=1 ; OPCODE FOR 'BASIC' MICRO INSTRUCTION
:751
:752 OPC1/=<22:20> ; MICRO OPCODE FIELD FOR THREE BIT OPCODES.
:753
:754 EXTENDED=2 ; OPCODE FOR 'EXTENDED' MICRO INSTRUCTION
:755 MOVE=3 ; OPCODE FOR 'MOVE' MICRO INSTRUCTION
:756
:757 OPC2/=<22:19> ; MICRO OPCODE FIELD FOR FOUR BIT OPCODES
:758
:759 MEM.REQ=3 ; OPCODE FOR 'MEM.REQ' MICRO INSTRUCTION
:760 MISC=2 ; OPCODE FOR 'MISC' MICRO INSTRUCTION
:761 JUMP=1 ; OPCODE FOR 'JUMP' MICRO INSTRUCTION
:762 DECODE=0 ; OPCODE FOR 'DECODE' MICRO INSTRUCTION
:763
:764
:765 ; THE FOLLOWING FOUR FIELDS ARE RELATED TO ADDRESSING
:766 ; VARIOUS RAM MEMORIES WITHIN THE DATA PATH.
:767
:768 ; THE A.ADRS REFERS TO THE 2901 INTERNAL RAM 'A-PORT'
    
```

```

:769 : THE B.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:770 : THE XB.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
:771 : THE D.ADRS REFERS TO THE LOWER 128 LOCATIONS OF THE LOCAL STORE RAM
:772 : THE XD.ADRS REFERS TO ALL 256 LOCATIONS OF THE LOCAL STORE RAM
:773
:774 : THE VALIDITY STATEMENTS ALLOW USAGE OF THESE FIELDS IN THE FOLLOWING MANNER
:775
:776 :     A.ADRS      EXTENDED MICRO INSTRUCTION
:777
:778 :     B.ADRS      BASIC MICRO INSTRUCTION
:779 :                 EXTENDED MICRO INSTRUCTION
:780 :                 MOVE MICRO INSTRUCTION
:781 :                 DECODE.IS MICRO INSTRUCTION
:782
:783 :     XB.ADRS      MOVE XWR MICRO INSTRUCTION
:784
:785 :     D.ADRS      BASIC MICRO INSTRUCTION
:786 :                 MEM.REQ MICRO INSTRUCTION
:787
:788 :     XD.ADRS      MOVE MICRO INSTRUCTION
:789
:790 : A.ADRS/=<10:9>,.VALIDITY=<A.VAL>
:791
:792 :     MASK=3      : REGISTER BACKUP MASK
:793
:794 : B.ADRS/=<8:7>,.VALIDITY=<B.VAL>,.DEFAULT=0
:795
:796 :     MASK=3      : REGISTER BACKUP MASK
:797
:798 : THIS ADDRESS FIELD WILL CONTAIN THE LOW TWO BITS OF THE 2901 B ADDRESS
:799 : THE THIRD BIT WILL BE FORCED BY HARDWARE.
:800
:801 : XB.ADRS/=<8:7>,.VALIDITY=<.EQL[<OPC1/>,<OPC1/MOVE>]>
:802
:803 :     BPC=0      : ADDRESS OF BEGINNING PC      WR[4]
:804 :     VA=1       : ADDRESS OF MEMORY REFERENCE WR[5]
:805 :     VAR=1      : ADDRESS OF LAST MEMORY REFERENCE. (WR[5] IN 2901).
:806
:807 : D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
:808
:809
:810 :     SAVED.2ND.REQUEST=0 : MM SAVED STATE FOR REQUEST.
:811 :     T1=1             : TEMP LOCATION 1
:812 :     T2=2             : TEMP LOCATION 2
:813 :     T3=3             : TEMP LOCATION 3
:814 :     T4=4             : TEMP LOCATION 4
:815 :     T5=5             : TEMP LOCATION 5
:816 :     T6=6             : TEMP LOCATION 6
:817 :     T7=7             : TEMP LOCATION 7
:818 :     T8=8             : TEMP LOCATION 8
:819 :     T9=9             : TEMP LOCATION 9
:820 :     T10=0A          : TEMP LOCATION 10
:821 :     BACKUP.CM.PC=0A : COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
:822
:823 :     T11=0B          : TEMP LOCATION 11
    
```

```

:824 : T12=0C : TEMP LOCATION 12
:825 : T13=0D : TEMP LOCATION 13
:826 : T14=0E : TEMP LOCATION 14
:827 : T15=0F : TEMP LOCATION 15
:828 : PC=10 : MACRO PROGRAM COUNTER
:829 : WR.BACKUP=11 : BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[]
:830 : SAVED.VAR=12 : MM SAVED STATE FOR VAR
:831 : SAVED.DATA=13 : MM SAVED STATE FOR DATA
:832 : SAVED.PTE.ADDRESS=14 : MM SAVED STATE FOR PAGE TABLE ENTRIES ADDRESS
:833 : SAVED.PTE=15 : MM SAVED STATE FOR PAGE TABLE ENTRIES
:834 : T16=16 : TEMP LOCATION 16
:835 : T17=17 : TEMP LOCATION 17
:836 : IE.TEMP4=18 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
:837 : #FFFFFF00=19 : CONSTANT
:838 : #FFFFFF00=1A : CONSTANT
:839 : #FFFFFFFC=1B : CONSTANT
:840 : #FFFFFFF=1C : CONSTANT

```

THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE
 HARDWARE VERSION OF THE PSL IS CHANGED (PSL.HW), THEN THIS
 CHANGE IS ALSO REPORTED HERE BY THE MICROCODE. ALL BITS OF
 THE PSL ARE REALLY LIKE THOSE IN THIS WORD EXCEPT FOR THE
 CONDITION CODES WHICH MUST BE OBTAINED FROM PSL.CC

```

:848 : PSL=1D : SOFT COPY OF VAX PSL
:849 : VAR=1E : VIRTUAL ADDRESS REGISTER
:850 : VAR2=1F : VIRTUAL ADDRESS REGISTER

```

```

:853 : #1=20 : MASK 00000001 (HEX)
:854 : #2=21 : MASK 00000002
:855 : #4=22 : MASK 00000004
:856 : #8=23 : MASK 00000008
:857 : #10=24 : MASK 00000010
:858 : #20=25 : MASK 00000020
:859 : #40=26 : MASK 00000040
:860 : #80=27 : MASK 00000080
:861 : #100=28 : MASK 00000100
:862 : #200=29 : MASK 00000200
:863 : #400=2A : MASK 00000400
:864 : #800=2B : MASK 00000800
:865 : #1000=2C : MASK 00001000
:866 : #2000=2D : MASK 00002000
:867 : #4000=2E : MASK 00004000
:868 : #8000=2F : MASK 00008000
:869 : #10000=30 : MASK 00010000
:870 : #20000=31 : MASK 00020000
:871 : #40000=32 : MASK 00040000
:872 : #80000=33 : MASK 00080000
:873 : #100000=34 : MASK 00100000
:874 : #200000=35 : MASK 00200000
:875 : #400000=36 : MASK 00400000
:876 : #800000=37 : MASK 00800000
:877 : #1000000=38 : MASK 01000000
:878 : #2000000=39 : MASK 02000000

```


:879	:	#40000000=3A	:	MASK 04000000
:880	:	#80000000=3B	:	MASK 08000000
:881	:	#100000000=3C	:	MASK 10000000
:882	:	#200000000=3D	:	MASK 20000000
:883	:	#400000000=3E	:	MASK 40000000
:884	:	#800000000=3F	:	MASK 80000000
:885	:		:	
:886	:	BIT0=20	:	MASK 00000001
:887	:	BIT1=21	:	MASK 00000002
:888	:	BIT2=22	:	MASK 00000004
:889	:	BIT3=23	:	MASK 00000008
:890	:	BIT4=24	:	MASK 00000010
:891	:	BIT5=25	:	MASK 00000020
:892	:	BIT6=26	:	MASK 00000040
:893	:	BIT7=27	:	MASK 00000080
:894	:	BIT8=28	:	MASK 00000100
:895	:	BIT9=29	:	MASK 00000200
:896	:	BIT10=2A	:	MASK 00000400
:897	:	BIT11=2B	:	MASK 00000800
:898	:	BIT12=2C	:	MASK 00001000
:899	:	BIT13=2D	:	MASK 00002000
:900	:	BIT14=2E	:	MASK 00004000
:901	:	BIT15=2F	:	MASK 00008000
:902	:	BIT16=30	:	MASK 00010000
:903	:	BIT17=31	:	MASK 00020000
:904	:	BIT18=32	:	MASK 00040000
:905	:	BIT19=33	:	MASK 00080000
:906	:	BIT20=34	:	MASK 00100000
:907	:	BIT21=35	:	MASK 00200000
:908	:	BIT22=36	:	MASK 00400000
:909	:	BIT23=37	:	MASK 00800000
:910	:	BIT24=38	:	MASK 01000000
:911	:	BIT25=39	:	MASK 02000000
:912	:	BIT26=3A	:	MASK 04000000
:913	:	BIT27=3B	:	MASK 08000000
:914	:	BIT28=3C	:	MASK 10000000
:915	:	BIT29=3D	:	MASK 20000000
:916	:	BIT30=3E	:	MASK 40000000
:917	:	BIT31=3F	:	MASK 80000000
:918	:		:	
:919	:	GPR0=40	:	GPR 0
:920	:	GPR1=41	:	GPR 1
:921	:	GPR2=42	:	GPR 2
:922	:	GPR3=43	:	GPR 3
:923	:	GPR4=44	:	GPR 4
:924	:	GPR5=45	:	GPR 5
:925	:	GPR6=46	:	GPR 6
:926	:	CM.SP=46	:	IN COMPATIBILITY THIS IS THE STACK POINTER.
:927	:	GPR7=47	:	GPR 7
:928	:	CM.PC=47	:	IN COMPATIBILITY MODE THIS IS THE PC.
:929	:	GPR8=48	:	GPR 8
:930	:	GPR9=49	:	GPR 9
:931	:	GPR10=4A	:	GPR 10
:932	:	GPR11=4B	:	GPR 11
:933	:	GPR12=4C	:	GPR 12

:934	: AP=4C	: ARGUMENT POINTER
:935	: GPR13=4D	: GPR 13
:936	: FP=4D	: FRAME POINTER
:937	: SP=4E	: GPR 14
:938	: TO=4F	: TEMP LOCATION 0
:939		
:940	: ZERO=50	: ZERO CONSTANT
:941	: #3=51	: CONSTANT
:942	: #5=52	: CONSTANT
:943	: #6=53	: CONSTANT
:944	: #7=54	: CONSTANT
:945	: #9=55	: CONSTANT
:946	: #B=56	: CONSTANT
:947	: #C=57	: CONSTANT
:948	: #D=58	: CONSTANT
:949	: #E=59	: CONSTANT
:950	: #F=5A	: CONSTANT
:951	: #13=5B	: CONSTANT
:952	: #1F=5C	: CONSTANT
:953	: #34=5D	: CONSTANT
:954	: #3B=5E	: CONSTANT
:955	: #3F=5F	: CONSTANT
:956	: #4B=60	: CONSTANT
:957	: #66=61	: CONSTANT
:958	: #A0=62	: CONSTANT
:959	: #F0=63	: CONSTANT
:960	: #FF=64	: CONSTANT
:961	: #1FF=65	: CONSTANT
:962	: #FFF=66	: CONSTANT
:963	: #2001=67	: CONSTANT
:964	: #3000=68	: CONSTANT
:965	: #7F80=69	: CONSTANT
:966	: #C000=6A	: CONSTANT
:967	: #FFE0=6B	: CONSTANT
:968	: #FFFF=6C	: CONSTANT
:969	: #1F0000=6D	: CONSTANT
:970	: #200001=6E	: CONSTANT
:971	: #F27000=6F	: CONSTANT
:972	: #3000000=70	: CONSTANT
:973	: #41F0000=71	: CONSTANT
:974	: #7000000=72	: CONSTANT
:975	: #3FFFFFF00=73	: CONSTANT
:976	: #7F800000=74	: CONSTANT
:977	: #7FFFFFF00=75	: CONSTANT
:978	: #7FFFFFF0E=76	: CONSTANT
:979	: #BF800001=77	: CONSTANT
:980		
:981	: GPR.OS=78	: HARDWARE INDEX TO GPRS
:982	: BIT.OS(3-0)=79	: 16 LOCATION HARDWARE INDEX TO BIT MASKS
:983	: BIT.OS(4-0)=7B	: 32 LOCATION HARDWARE INDEX TO BIT MASKS
:984	: OS=7C	: OS REGISTER
:985	: DEC.CON=7D	: DECIMAL CARRIES
:986	: SIZE=7E	: SIZE REGISTER
:987	: MDT= 7E	: MEMORY REFERENCE DATA SIZE
:988	: INTERRUPT.VEC=7E	: HARDWARE INTERRUPT VECTOR

```

:989 :
:990 : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:991 : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:992 :
:993 : PSL.CC=7F : PSL CONDITION CODES
:994 :
:995 : XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
:996 :
:997 : SAVED.2ND.REQUEST=0 : MM SAVED STATE FOR REQUEST.
:998 : T1=1 : TEMP LOCATION 1
:999 : T2=2 : TEMP LOCATION 2
:1000 : T3=3 : TEMP LOCATION 3
:1001 : T4=4 : TEMP LOCATION 4
:1002 : T5=5 : TEMP LOCATION 5
:1003 : T6=6 : TEMP LOCATION 6
:1004 : T7=7 : TEMP LOCATION 7
:1005 : T8=8 : TEMP LOCATION 8
:1006 : T9=9 : TEMP LOCATION 9
:1007 : T10=0A : TEMP LOCATION 10
:1008 : BACKUP.CM.PC=0A : COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
:1009 : T11=0B : TEMP LOCATION 11
:1010 : T12=0C : TEMP LOCATION 12
:1011 : T13=0D : TEMP LOCATION 13
:1012 : T14=0E : TEMP LOCATION 14
:1013 : T15=0F : TEMP LOCATION 15
:1014 : PC=10 : MACRO INSTRUCTION PC
:1015 : WR.BACKUP=11 : BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[].
:1016 : SAVED.VAR=12 : MM SAVED VAR LOCATION
:1017 : SAVED.DATA=13 : MM SAVED DATA LOCATION
:1018 : SAVED.PTE.ADDRESS=14 : MM SAVED PAGE TABLE ENTRY ADDRESS
:1019 : SAVED.PTE=15 : MM SAVED PAGE TABLE ENTRY
:1020 : T16=16 : TEMP LOCATION 16
:1021 : T17=17 : TEMP LOCATION 17
:1022 : IE.TEMP4=18 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
:1023 : #FFFFFF00=19 : CONSTANT
:1024 : #FFFFFF00=1A : CONSTANT
:1025 : #FFFFFFFC=1B : CONSTANT
:1026 : #FFFFFFFF=1C : CONSTANT
:1027 :
:1028 : THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL. WHENEVER THE HARDWARE
:1029 : VERSION OF THE PSL IS CHANGED (PSL.HW) THEN THIS CHANGE IS
:1030 : REPORTED HERE. ALL BITS OF THE PSL ARE REALLY LIKE THOSE HERE
:1031 : EXCEPT FOR THE CONDITION CODES WHICH MUST BE OBTAINED
:1032 : FROM PSL.CC.
:1033 :
:1034 : PSL=1D : SOFT COPY OF VAX PSL
:1035 : VAR=1E : VIRTUAL ADDRESS REGISTER
:1036 : VAR2=1F : VIRTUAL ADDRESS REGISTER
:1037 :
:1038 :
:1039 : #1=20 : MASK 00000001 (HEX)
:1040 : #2=21 : MASK 00000002
:1041 : #4=22 : MASK 00000004
:1042 : #8=23 : MASK 00000008
:1043 : #10=24 : MASK 00000010
    
```


:1044	: #20=25	: MASK 00000020
:1045	: #40=26	: MASK 00000040
:1046	: #80=27	: MASK 00000080
:1047	: #100=28	: MASK 00000100
:1048	: #200=29	: MASK 00000200
:1049	: #400=2A	: MASK 00000400
:1050	: #800=2B	: MASK 00000800
:1051	: #1000=2C	: MASK 00001000
:1052	: #2000=2D	: MASK 00002000
:1053	: #4000=2E	: MASK 00004000
:1054	: #8000=2F	: MASK 00008000
:1055	: #10000=30	: MASK 00010000
:1056	: #20000=31	: MASK 00020000
:1057	: #40000=32	: MASK 00040000
:1058	: #80000=33	: MASK 00080000
:1059	: #100000=34	: MASK 00100000
:1060	: #200000=35	: MASK 00200000
:1061	: #400000=36	: MASK 00400000
:1062	: #800000=37	: MASK 00800000
:1063	: #1000000=38	: MASK 01000000
:1064	: #2000000=39	: MASK 02000000
:1065	: #4000000=3A	: MASK 04000000
:1066	: #8000000=3B	: MASK 08000000
:1067	: #10000000=3C	: MASK 10000000
:1068	: #20000000=3D	: MASK 20000000
:1069	: #40000000=3E	: MASK 40000000
:1070	: #80000000=3F	: MASK 80000000
:1071		
:1072	: BIT0=20	: MASK 00000001
:1073	: BIT1=21	: MASK 00000002
:1074	: BIT2=22	: MASK 00000004
:1075	: BIT3=23	: MASK 00000008
:1076	: BIT4=24	: MASK 00000010
:1077	: BIT5=25	: MASK 00000020
:1078	: BIT6=26	: MASK 00000040
:1079	: BIT7=27	: MASK 00000080
:1080	: BIT8=28	: MASK 00000100
:1081	: BIT9=29	: MASK 00000200
:1082	: BIT10=2A	: MASK 00000400
:1083	: BIT11=2B	: MASK 00000800
:1084	: BIT12=2C	: MASK 00001000
:1085	: BIT13=2D	: MASK 00002000
:1086	: BIT14=2E	: MASK 00004000
:1087	: BIT15=2F	: MASK 00008000
:1088	: BIT16=30	: MASK 00010000
:1089	: BIT17=31	: MASK 00020000
:1090	: BIT18=32	: MASK 00040000
:1091	: BIT19=33	: MASK 00080000
:1092	: BIT20=34	: MASK 00100000
:1093	: BIT21=35	: MASK 00200000
:1094	: BIT22=36	: MASK 00400000
:1095	: BIT23=37	: MASK 00800000
:1096	: BIT24=38	: MASK 01000000
:1097	: BIT25=39	: MASK 02000000
:1098	: BIT26=3A	: MASK 04000000

:1099	:	BIT27=3B	:	MASK 08000000
:1100	:	BIT28=3C	:	MASK 10000000
:1101	:	BIT29=3D	:	MASK 20000000
:1102	:	BIT30=3E	:	MASK 40000000
:1103	:	BIT31=3F	:	MASK 80000000
:1104	:			
:1105	:	GPR0=40	:	GPR 0
:1106	:	GPR1=41	:	GPR 1
:1107	:	GPR2=42	:	GPR 2
:1108	:	GPR3=43	:	GPR 3
:1109	:	GPR4=44	:	GPR 4
:1110	:	GPR5=45	:	GPR 5
:1111	:	GPR6=46	:	GPR 6
:1112	:	CM.SP=46	:	IN COMPATIBILITY MODE THIS IS THE STACK POINTER.
:1113	:	GPR7=47	:	GPR 7
:1114	:	CM.PC=47	:	IN COMPATIBILITY MODE THIS IS THE PC.
:1115	:	GPR8=48	:	GPR 8
:1116	:	GPR9=49	:	GPR 9
:1117	:	GPR10=4A	:	GPR 10
:1118	:	GPR11=4B	:	GPR 11
:1119	:	GPR12=4C	:	GPR 12
:1120	:	AP=4C	:	ARGUMENT POINTER
:1121	:	GPR13=4D	:	GPR 13
:1122	:	FP=4D	:	FRAME POINTER
:1123	:	SP=4E	:	GPR 14
:1124	:	TO=4F	:	TEMP LOCATION 0
:1125	:			
:1126	:	ZERO=50	:	ZERO CONSTANT
:1127	:	#3=51	:	CONSTANT
:1128	:	#5=52	:	CONSTANT
:1129	:	#6=53	:	CONSTANT
:1130	:	#7=54	:	CONSTANT
:1131	:	#9=55	:	CONSTANT
:1132	:	#B=56	:	CONSTANT
:1133	:	#C=57	:	CONSTANT
:1134	:	#D=58	:	CONSTANT
:1135	:	#E=59	:	CONSTANT
:1136	:	#F=5A	:	CONSTANT
:1137	:	#13=5B	:	CONSTANT
:1138	:	#1F=5C	:	CONSTANT
:1139	:	#34=5D	:	CONSTANT
:1140	:	#3B=5E	:	CONSTANT
:1141	:	#3F=5F	:	CONSTANT
:1142	:	#4B=60	:	CONSTANT
:1143	:	#66=61	:	CONSTANT
:1144	:	#A0=62	:	CONSTANT
:1145	:	#F0=63	:	CONSTANT
:1146	:	#FF=64	:	CONSTANT
:1147	:	#1FF=65	:	CONSTANT
:1148	:	#FFF=66	:	CONSTANT
:1149	:	#2001=67	:	CONSTANT
:1150	:	#3000=68	:	CONSTANT
:1151	:	#7F80=69	:	CONSTANT
:1152	:	#C000=6A	:	CONSTANT
:1153	:	#FFE0=6B	:	CONSTANT

```

:1154 : #FFFF=6C : CONSTANT
:1155 : #1F0000=6D : CONSTANT
:1156 : #200001=6E : CONSTANT
:1157 : #F27000=6F : CONSTANT
:1158 : #3000000=70 : CONSTANT
:1159 : #41F0000=71 : CONSTANT
:1160 : #7000000=72 : CONSTANT
:1161 : #3FFFFFF00=73 : CONSTANT
:1162 : #7F800000=74 : CONSTANT
:1163 : #7FFFFFF00=75 : CONSTANT
:1164 : #7FFFFFF0E=76 : CONSTANT
:1165 : #BF800001=77 : CONSTANT
:1166 :
:1167 : GPR.OS=78 : HARDWARE INDEX TO GPRS
:1168 : BIT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO BIT MASKS
:1169 : BIT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO BIT MASKS
:1170 : OS=7C : OS REGISTER
:1171 : DEC.CON=7D : DECIMAL CARRIES
:1172 : SIZE=7E : SIZE REGISTER
:1173 : MDT= 7E : MEMORY REFERENCE DATA SIZE
:1174 : INTERRUPT.VEC=7E : HARDWARE INTERRUPT VECTOR
:1175 :
:1176 : THIS LOCATION IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE
:1177 : READ OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:1178 :
:1179 : PSL.CC=7F : PSL CONDITION CODES
:1180 :
:1181 : THESE ADDRESSES ARE ONLY ACCESSIBLE WITH THE MOV MICRO INSTRUCTION.
:1182 :
:1183 : KSP=80 : KERNEL STACK POINTER
:1184 : ESP=81 : EXECUTIVE STACK POINTER
:1185 : SSP=82 : SUPERVISOR STACK POINTER
:1186 : USP=83 : USER STACK POINTER
:1187 : ISP=84 : INTERRUPT STACK POINTER
:1188 : ASTLVL=85 : AST LEVEL
:1189 : PCBB=86 : PROCESS CONTROL BLOCK BASE
:1190 : SCBB=87 : SYSTEM CONTROL BLOCK BASE
:1191 : SISR=88 : SOFTWARE INTERRUPT SUMMARY REGISTER
:1192 :
:1193 : CONSOLE.COMMAND=89 : ONE BYTE COMMAND.
:1194 : CONSOLE.MODIFIER=8A : ONE BYTE MODIFIER.
:1195 : CONSOLE.ADDRESS=8B : FOUR BYTES OF ADDRESS.
:1196 : CONSOLE.DATA=8C : FOUR BYTES OF DATA.
:1197 : CONSOLE.STATUS=8D : ONE BYTE OF STATUS.
:1198 : PSEUDO.NICR=8E : COPY OF NICR THAT 8085 HAS.
:1199 : PSEUDO.ICR=8F : COPY OF ICR THAT 8085 HAS.
:1200 : DEBUG.SAVE.WR0=90 : TEMPORARY SAVE WR0.
:1201 : DEBUG.SAVE.WR1=91 : TEMPORARY SAVE WR1.
:1202 : DEBUG.SAVE.ALU.CC=92 : TEMPORARY SAVE CONDITION CODES.
:1203 : MEMORY.SIZE=93 : MEMORY SIZE OF THE MACHINE.
:1204 : CRD.LOG=94 : INFORMATION ABOUT LAST SOFT MEMORY ERROR.
:1205 : PACKET.A=95 : TOP OF PACKET.
:1206 : PACKET.B=96 : BOTTOM OF PACKET.
:1207 : DEBUG.SAVE.ADDRESS=97 : ADDRESS FOR EXAMINE/DEPOSIT.
:1208 :
    
```


:1209 : SAVED.MDT=99 : SAVED ADDRESS OF MDT FOR MMIE OPTIMIZATION.
 :1210 : POBR=9A : P0 BASE REGISTER
 :1211 : POLR=9B : P0 LENGTH REGISTER
 :1212 : P1BR=9C : P1 BASE REGISTER
 :1213 : P1LR=9D : P1 LENGTH REGISTER
 :1214 : SBR=9E : SYSTEM BASE REGISTER
 :1215 : SLR=9F : SYSTEM LENGTH REGISTER

:1216 :
 :1217 : PORT0=0A0 : RESERVED FOR PORT.
 :1218 : PORT1=0A1 : RESERVED FOR PORT.
 :1219 : PORT2=0A2 : RESERVED FOR PORT.
 :1220 : PORT3=0A3 : RESERVED FOR PORT.
 :1221 : PORT4=0A4 : RESERVED FOR PORT.
 :1222 : PORT5=0A5 : RESERVED FOR PORT.
 :1223 : PORT6=0A6 : RESERVED FOR PORT.
 :1224 : PORT7=0A7 : RESERVED FOR PORT.
 :1225 : PORT8=0A8 : RESERVED FOR PORT.
 :1226 : PORT9=0A9 : RESERVED FOR PORT.
 :1227 : PORT10=0AA : RESERVED FOR PORT.
 :1228 : PORT11=0AB : RESERVED FOR PORT.
 :1229 : PORT12=0AC : RESERVED FOR PORT.
 :1230 : PORT13=0AD : RESERVED FOR PORT.
 :1231 : PORT14=0AE : RESERVED FOR PORT.
 :1232 : PORT15=0AF : RESERVED FOR PORT.
 :1233 : PORT16=0B0 : RESERVED FOR PORT.
 :1234 : PORT17=0B1 : RESERVED FOR PORT.
 :1235 : PORT18=0B2 : RESERVED FOR PORT.
 :1236 : PORT19=0B3 : RESERVED FOR PORT.
 :1237 : PORT20=0B4 : RESERVED FOR PORT.
 :1238 : PORT21=0B5 : RESERVED FOR PORT.
 :1239 : PORT22=0B6 : RESERVED FOR PORT.
 :1240 : PORT23=0B7 : RESERVED FOR PORT.
 :1241 : PORT24=0B8 : RESERVED FOR PORT.
 :1242 : PORT25=0B9 : RESERVED FOR PORT.
 :1243 : PORT26=0BA : RESERVED FOR PORT.
 :1244 : PORT27=0BB : RESERVED FOR PORT.
 :1245 : PORT28=0BC : RESERVED FOR PORT.
 :1246 : PORT29=0BD : RESERVED FOR PORT.
 :1247 : PORT30=0BE : RESERVED FOR PORT.
 :1248 : PORT31=0BF : RESERVED FOR PORT.

:1249 :
 :1250 : XGPR0=0C0 : BACKUP COPY OF GPR 0
 :1251 : XGPR1=0C1 : BACKUP COPY OF GPR 1
 :1252 : XGPR2=0C2 : BACKUP COPY OF GPR 2
 :1253 : XGPR3=0C3 : BACKUP COPY OF GPR 3
 :1254 : XGPR4=0C4 : BACKUP COPY OF GPR 4
 :1255 : XGPR5=0C5 : BACKUP COPY OF GPR 5
 :1256 : XGPR6=0C6 : BACKUP COPY OF GPR 6
 :1257 : XGPR7=0C7 : BACKUP COPY OF GPR 7
 :1258 : XGPR8=0C8 : BACKUP COPY OF GPR 8
 :1259 : XGPR9=0C9 : BACKUP COPY OF GPR 9
 :1260 : XGPR10=0CA : BACKUP COPY OF GPR 10
 :1261 : XGPR11=0CB : BACKUP COPY OF GPR 11
 :1262 : XGPR12=0CC : BACKUP COPY OF GPR 12
 :1263 : XGPR13=0CD : BACKUP COPY OF GPR 13

```

:1264 : XSP=0CE : BACKUP COPY OF GPR 14
:1265 :
:1266 : #14=0D0 : CONSTANT
:1267 : #18=0D1 : CONSTANT
:1268 : #1C=0D2 : CONSTANT
:1269 : #24=0D3 : CONSTANT
:1270 : #28=0D4 : CONSTANT
:1271 : #2C=0D5 : CONSTANT
:1272 : #2D=0D6 : CONSTANT
:1273 : #30=0D7 : CONSTANT
:1274 : #F8=0D8 : CONSTANT
:1275 : #FC=0D9 : CONSTANT
:1276 : #2D20=0DA : CONSTANT
:1277 : #140000=0DB : CONSTANT
:1278 : #150000=0DC : CONSTANT
:1279 : #160000=0DD : CONSTANT
:1280 : #170000=0DE : CONSTANT
:1281 : #180000=0DF : CONSTANT
:1282 : #1E0000=0E0 : CONSTANT
:1283 : #40A10=0E1 : CONSTANT
:1284 : #C0000E0=0E2 : CONSTANT
:1285 : #0FFF0000=0E3 : CONSTANT
:1286 : #B020FF00=0E4 : CONSTANT
:1287 : #FFFFFF10=0E5 : CONSTANT
:1288 : DEBUG.SAVE.T1=0E6 : TEMPORARY SAVE T1.
:1289 : DEBUG.SAVE.OS=0E7 : TEMPORARY SAVE OS.
:1290 : DEBUG.SAVE.SIZE=0E8 : TEMPORARY SAVE SIZE VALUE.
:1291 : SAVED.WR0=0E9 : MM SAVED WR0
:1292 : SAVED.WR1=0EA : MM SAVED WR1
:1293 : SAVED.2ND.VAR=0EB : MM SAVED VAR 2
:1294 : SAVED.ALU.CC=0EC : MM SAVED ALU CONDITION CODES
:1295 : SAVED.SIZE=0ED : MM SAVED LS[SIZE]
:1296 : SAVED.OS=0EE : MM SAVED LS[OS]
:1297 : SAVED.REQUEST=0EF : MM SAVED REQUEST DATA
:1298 : SAVED.CRD.LOG=0F0 : MM SAVED LOCATION
:1299 : OS.BAK=0F1 : USED BY WARM FLOATING POINT.
:1300 :
:1301 : THIS LOCATION IS FOR USE BY THE MFPR AND MTPR TO THE CONSOLE AND
:1302 : TU58 CSR'S. ALL OF THE INTERRUPT ENABLE BITS ARE HEREIN.
:1303 :
:1304 : CONSOLE/PROGRAM I/O MODE FLAG - BIT<31> 0 - CONSOLE 1 - PROGRAM I/O MODE
:1305 :
:1306 : CRD RETRY IN PROGRESS - BIT<5> 0 - NOTHING DOING 1 - IN PROGRESS
:1307 : MACHINE CHECK IN PROGRESS - BIT<4>
:1308 : TU58 TRANSMIT CSR IE - BIT<3>
:1309 : TU58 RECEIVER CSR IE - BIT<2>
:1310 : CONSOLE TRANSMIT CSR IE - BIT<1>
:1311 : CONSOLE TRANSMIT CSR IE - BIT<0>
:1312 :
:1313 : CONSOLE.CSR.IES=0F2 : CONSOLE CSR IE'S AND CONSOLE/PROGRAM I/O MODE FLAG.
:1314 : IE.TEMP1=0F3 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION ONE
:1315 : IE.TEMP2=0F4 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION TWO
:1316 : IE.TEMP3=0F5 : INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION THREE
:1317 :
:1318 : XGPR.OS=0F8 : HARDWARE INDEX TO XGPRS
    
```

:1319 : PORT(3-0)=0F9 : 16 LOCATION HARDWARE INDEX TO PORT REGISTERS.
 :1320 : PORT(4-0)=0FB : 32 LOCATION HARDWARE INDEX TO PORT REGISTERS.
 :1321 : CCR=0FC : CONSOLE READ REGISTER
 :1322 : TIMER=0FD : INTERVAL TIMER VALUE.
 :1323 : ALU.CC=0FE : ALU CONDITION CC'S

:1324 :
 :1325 : THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
 :1326 : PSL ARE AFFECTED:

:1327 :
 :1328 : COMPATIBILITY MODE - BIT<31>
 :1329 : CURRENT MODE - BITS<25:24>
 :1330 : INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
 :1331 : TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
 :1332 : NEGATIVE CONDITION CODE - BIT<3>
 :1333 : ZERO CONDITION CODE - BIT<2>
 :1334 : OVERFLOW CONDITION CODE - BIT<1>
 :1335 : CARRY CONDITION CODE - BIT<0>

:1336 :
 :1337 : PSL.HW=OFF : HARDWARE PSL BITS

:1338 :
 :1339 : DATA PATH CONTROL

:1340 :
 :1341 : THIS FIELD IS BUILT FROM AN ADDRESS IN THE CCODE MEMORY. THE CCODE MEMORY
 :1342 : CONTAINS DATA PATH MICRO INSTRUCTIONS TO PRODUCE THE DESIRED OPERATION
 :1343 : FOR THE UCODE MACROS.
 :1344 :

:1345 : DATA PATH CONTROL FIELD FOR THE 'BASIC' MICRO INSTRUCTION

:1346 :
 :1347 : DP/= <21:16>, .VALIDITY=<DP.VAL>, .DEFAULT=<.SHIFT[.AND[<SDP/Q.CMP.LS>,OFF],-2]>

:1348 :
 :1349 : THIS FIELD IS ONLY USED WITH THE INSTRUCTIONS WHICH CAN HAVE
 :1350 : XCHG ATTACHED.

:1351 :
 :1352 : DP.SMALL/= <20:16>, .VALIDITY=<DP.VAL>

:1353 :
 :1354 : EXCHANGE ORIGINAL WR TO LS

:1355 :
 :1356 : XCHG.BIT/= <21>, .DEFAULT=<0>

:1357 :
 :1358 : YES=1
 :1359 : NO=0

:1360 : DATA PATH CONTROL FIELD FOR THE 'EXTENDED' MICRO INSTRUCTION

:1361 :
 :1362 : XDP/= <19:14>, .VALIDITY=<A.VAL>

:1363 : DATA PATH CONTROL FOR THE 'MOVE' MICRO INSTRUCTION

:1364 :
 :1365 : MDP/= <19:17>, .VALIDITY=<XD.VAL>

:1366 :


```

:1370 .PAGE
:1371 : CONDITION CODE FIELD / DATA PATH CONTEXT
:1372
:1373 CC/= <6:5> , .VALIDITY= <CC.VAL> , .DEFAULT= <CC/DT(LONG)&HOLD.ALU.CC>
:1374
:1375 DT(LONG)&HOLD.ALU.CC=0 : USE LONG CONTEXT(32 BITS) AND HOLD ALU CONDITION CODES
:1376 DT(LONG)&SET.ALU.CC=1 : USE LONG CONTEXT(32 BITS) AND SET ALU CONDITION CODES
:1377 DT(SIZE)&SET.ALU.CC=2 : USE CONTEXT IN THE SIZE REGISTER AND SET ALU CONDITION CODES
:1378 DT(SIZE)&MAP.ALU.CC.TO.PSL=3 : TRANSFER ALU CONDITION CODES TO PSL CONDITION CODES HARDWARE DEPENDENT
:1379
:1380
:1381 : MEMORY DATA TYPE FIELD
:1382
:1383 MDT/= <6:5> , .VALIDITY= <DT.VAL>
:1384
:1385 BYTE=0 : SIZE = BYTE
:1386 WORD=1 : SIZE = WORD
:1387 SIZE=2 : SIZE = CONTENTS OF SIZE REGISTER
:1388 LONG=3 : SIZE = LONG
:1389
:1390
:1391 : SHIFT INPUT FIELD
:1392
:1393 : THIS CONTROL FIELD DETERMINES THE SHIFT INPUTS FOR THE FOLLOWING OPERATIONS.
:1394 : THE DIRECTION OF THE SHIFT IS DETERMINED BY THE 2901 DESTINATION CODE.
:1395
:1396 SI/= <13:11> , .VALIDITY= <A.VAL>
:1397
:1398 ROT.WR=0 : ROTATE WR
:1399 ROT.WR.Q=1 : ROTATE WR AND Q
:1400 ASH.WR=2 : ARITHMETIC SHIFT OF WR
:1401 ASH.WR.Q=3 : ARITHMETIC SHIFT OF WR AND Q
:1402 MUL.SHF=4 : SIGNED MULTIPLY SHIFT OPERATION
:1403 UMUL.SHF=5 : UNSIGNED MULTIPLY SHIFT OPERATION
:1404 SHF.WR.Q=6 : LOGICAL SHIFT WR AND Q
:1405
:1406
:1407 : SKIP MICRO CONTROL FIELD
:1408
:1409 : THIS FIELD IS VALID FOR ALL INSTRUCTIONS EXCEPT
:1410
:1411 JUMP MICRO INSTRUCTION
:1412 DECODE MICRO INSTRUCTION
:1413
:1414 SCTL/= <4:0> , .VALIDITY= <.AND[<SCTL.VAL>,<SKIP.PAGE.VAL>]> , .DEFAULT= <SCTL/NO.SKIP>
:1415
:1416 N.CLR=0 : ALU N-BIT EQUAL ZERO
:1417 NEQ=1 : ALU Z-BIT EQUAL ZERO
:1418 BIT.SET=1 : ALU Z-BIT EQUAL ZERO
:1419 V.CLR=2 : ALU V-BIT EQUAL ZERO
:1420 C.SET=3 : ALU C-BIT EQUAL ONE
:1421 GEQU=3 : ALU C-BIT EQUAL ONE
:1422 NOT.PSL.C=4 : PSL C EQUAL ZERO
:1423 BR.FALSE=5 : BRANCH INSTRUCTION FALSE
:1424 R.DST=6 : REGISTER MODE FLAG
  
```

:1425	NO.INTERRUPT=7	: NO INTERRUPT PENDING
:1426	N.SET=8	: ALU N-BIT EQUAL ONE
:1427	EQL=9	: ALU Z-BIT EQUAL ONE
:1428	BITS.CLR=9	: ALU Z-BIT EQUAL ONE
:1429	V.SET=0A	: ALU V-BIT EQUAL ONE
:1430	C.CLR=0B	: ALU C-BIT EQUAL ZERO
:1431	LSSU=0B	: ALU C-BIT EQUAL ZERO
:1432	STATE.ZERO.SET=0C	: STATE REGISTER BIT ZERO TRUE
:1433	STATE.ONE.SET=0D	: STATE REGISTER BIT ONE TRUE
:1434	STATE.ZERO.CLR=0E	: STATE REGISTER BIT ZERO FALSE
:1435	STATE.ONE.CLR=0F	: STATE REGISTER BIT ONE FALSE
:1436	RET.NOERR=10	: RETURN+1 IF NO MEMORY ERROR
:1437	JMP.Z.CLR=11	: JUMP TO (STACK) IF ALU Z = 0
:1438	POP.USTACK=12	: DISCARD TOP STACK ENTRY
:1439	JMP.C.SET=13	: JUMP TO (STACK) IF ALU C=1
:1440	RETURN=14	: RETURN TO (USTACK)
:1441	NO.SKIP=15	: EXECUTE NEXT INSTRUCTION
:1442	RETURN+1=16	: RETURN TO (USTACK)+1
:1443	LOOP.IF.NO.I.AND.SYNC=17	: JUMP TO (STACK) IF NO INTERRUPT AND NO ACCELERATOR SYNC.
:1444	CONSOLE.ATTN=18	: CONSOLE ATTENTION
:1445	CONSOLE.ACK=19	: CONSOLE ACKNOWLEDGE
:1446	NO.FAST.INTERRUPT=1A	: NO FAST INTERRUPTS PENDING
:1447	BACKUP.MASK.VALID=1B	: REGISTER BACKUP MASK VALID.
:1448	LSS=1C	: SIGNED LESS THAN.
:1449	MEM.REF.OK=1D	: NO MEMORY ERROR TRUE
:1450	SKIP=1E	: EXECUTE ADDRESS PLUS ONE
:1451	SYNC=1F	: ACCELERATOR SYNCHRONIZATION

:1452
:1453
:1454 : JUMP MICRO CONTROL FIELD

:1455
:1456 : THIS FIELD IS VALID FOR THE FOLLOWING INSTRUCTIONS

:1457
:1458 : JUMP MICRO INSTRUCTION
:1459 : DECODE MICRO INSTRUCTION

:1460
:1461 JCTL/= <3:0> ,.VALIDITY=<.AND[<JCTL.VAL>,<JUMP.PAGE.VAL>]> ,.DEFAULT=<JCTL/JUMP.VALID>

:1462	N.CLR=0	: ALU N-BIT EQUAL ZERO
:1463	NEQ=1	: ALU Z-BIT EQUAL ZERO
:1464	BIT.SET=1	: ALU Z-BIT EQUAL ZERO
:1465	V.CLR=2	: ALU V-BIT EQUAL ZERO
:1466	C.SET=3	: ALU C-BIT EQUAL ONE
:1467	GEQU=3	: ALU C-BIT EQUAL ONE
:1468	JUMP=4	: JUMP UNCONDITIONALLY
:1469	BR.FALSE=5	: BRANCH INSTRUCTION FALSE
:1470	R.DST=6	: REGISTER MODE FLAG
:1471	NO.INTERRUPT=7	: NO INTERRUPT PENDING
:1472	N.SET=8	: ALU N-BIT EQUAL ONE
:1473	EQL=9	: ALU Z-BIT EQUAL ONE
:1474	BITS.CLR=9	: ALU Z-BIT EQUAL ONE
:1475	V.SET=0A	: ALU V-BIT EQUAL ONE
:1476	C.CLR=0B	: ALU C-BIT EQUAL ZERO
:1477	LSSU=0B	: ALU C-BIT EQUAL ZERO
:1478	JSR=0C	: UNCONDITIONAL JUMP AND PUSH USTACK
:1479		

```

:1480      JSR.VALID=0D      ; JUMP AND PUSH USTACK IF IB VALID=1
:1481      NO.JUMP.TST=0E   ; DO NOT JUMP AND SKIP IF IB VALID=0
:1482      JUMP.VALID=0F    ; JUMP IF IB VALID=1
:1483
:1484
:1485      ; JUMP ADDRESS FIELD
:1486      ;
:1487      JUMP.ADRS/=<17:4>,.ADDRESS,.VALIDITY=<JUMP.VAL>
:1488
:1489
:1490      ; OS ENABLE
:1491      ;
:1492      OS/=<18>,.VALIDITY=<JUMP.VAL>
:1493
:1494      NOP=0
:1495      WITH.OS=1        ; CONCATENATE OS REGISTER TO JUMP ADDRESS
:1496
:1497
  
```



```

:1498 .PAGE
:1499 .BACKUP PC
:1500
:1501 BPC/= <18> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=<.CASE[<.EQL[<OPC2/> ,<OPC2/DECODE>]]>]OF[0,1]>
:1502
:1503     NOP=1
:1504     SAVE.PC=0           ; WRITE ALU DATA INTO WR[B] (PC+1)
:1505
:1506
:1507 ; DECODE ADDRESS FIELD
:1508
:1509 DEC.ADRS/= <17:12> ,.VALIDITY=<DECODE.VAL>
:1510
:1511
:1512 ; IR FUNCTION
:1513
:1514 IFUNC/= <11:9> ,.VALIDITY=<DECODE.VAL>
:1515
:1516     CM.EXEC=0           ; COMPATIBILITY MODE EXECUTION DISPATCH WHEN UPPER BYTE = ZERO
:1517     SPEC=0             ; SPECIFIER DISPATCH
:1518     CM.IRD=1           ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
:1519     FLOAT=1           ; SPECIFIER FLOAT TYPE DISPATCH
:1520     VAX.EXEC=2        ; VAX EXECUTION DISPATCH
:1521     ASRC=2            ; ADDRESS SPECIFIER DISPATCH
:1522     VAX.IRD=3        ; VAX OP CODE DISPATCH
:1523     VSRC=4           ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
:1524     ESRC=5           ; SPECIFIER DISPATCH IN INDEX MODE
:1525     CM.DST=6         ; COMPATIBILITY MODE DESTINATION DISPATCH
:1526     CM.SINGLE=7      ; COMPATIBILITY MODE SOP DISPATCH
:1527
:1528
:1529 ; INSTRUCTION BUFFER REQUEST
:1530
:1531 IB.REQ/= <8> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=0
:1532
:1533     NOP=0
:1534     IB.REQ=1          ; ENABLE HARDWARE DEPENDENT MEMORY REQUEST
:1535
:1536 ; COMPATIBILITY MODE OP CODE SELECT
:1537
:1538 CM.HI/= <7> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=0
:1539
:1540     LOW.BYTE=0        ; DECODE IR BITS<7:0>
:1541     HI.BYTE=1         ; DECODE IR BITS <15:6>
:1542
:1543
:1544 ; LOAD OS REGISTER
:1545
:1546 LD.OS/= <6> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=0
:1547
:1548     NOP=0
:1549     LOAD.OS=1          ; LOAD OS REGISTER FROM I-BUFFER
:1550
:1551
:1552 ; REGISTER DESTINATION FLAG ENABLE
    
```

```

:1553 :
:1554 R.DST/= <5> ,.VALIDITY=<DECODE.VAL> ,.DEFAULT=0
:1555
:1556 NOP=0
:1557 ENAB=1
:1558 :
:1559 : OPCODE SPECIFIER BIT
:1560 :
:1561 OPC.SPEC/= <4> ,.VALIDITY=<DECODE.VAL>
:1562
:1563 CM.EXEC=1 : COMPATIBILITY MODE EXECUTION DISPATCH
:1564 SPEC=0 : SPECIFIER DISPATCH
:1565 CM.IRD=1 : COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
:1566 FLOAT=0 : SPECIFIER FLOAT TYPE DISPATCH
:1567 VAX.EXEC=1 : VAX EXECUTION DISPATCH
:1568 ASRC=0 : ADDRESS SPECIFIER DISPATCH
:1569 VAX.IRD=1 : VAX OPCODE DISPATCH
:1570 VSRC=0 : ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
:1571 ESRC=0 : SPECIFIER DISPATCH IN INDEX MODE
:1572 CM.DST=0 : COMPATIBILITY MODE DESTINATION DISPATCH
:1573 CM.SINGLE=0 : COMPATIBILITY MODE SOP DISPATCH
  
```

```

:1574 .PAGE
:1575 . MISCELLANEOUS FUNCTIONS
:1576
:1577
:1578 . THIS FIELD IS A FLAG FOR STEERING THE MICROWORD AS A MISCELLANEOUS
:1579 . INSTRUCTION OR AS A PORT INSTRUCTION.
:1580
:1581 MISC.PORT/=<18>
:1582
:1583     MISC=0
:1584     PORT=1
:1585
:1586 MISC.0/=<17>,.VALIDITY=<MISC.VAL>
:1587
:1588     NOP=0 ; NO OPERATION IN ALU OR CONDITION CODES.
:1589
:1590 MISC.1/=<15:12>,.VALIDITY=<MISC.VAL>
:1591
:1592     NOP=0F ; NO OPERATION
:1593     SET.CP.ATTN=0E ; SET CPU ATTENTION FOR CONSOLE
:1594     SET.CP.ACK=0D ; SET CPU ACKNOWLEDGE FOR CONSOLE
:1595     CLR.CP.ATTN.AND.ACK=0C ; CLEAR CPU ATTENTION AND CPU ACKNOWLEDGE
:1596     SET.HIGH.PAGE=0B ; SET BIT FOR HIGH HALF OF WCS.
:1597     CLR.HIGH.PAGE=0A ; CLEAR BIT FOR LOW HALF OF WCS.
:1598     SET.PORT.I.O=9 ; SET PORT I/O MODE
:1599     SET.ACC.I.O=8 ; SET ACCELERATOR I/O MODE
:1600     SET.BACKUP.MASK.VALID=7 ; SET REGISTER BACKUP MASK FLAG
:1601     SET.S1=6 ; SET STATE ONE BIT
:1602     SET.S0=5 ; SET STATE ZERO BIT
:1603     CLR.S1=4 ; CLEAR STATE ONE BIT
:1604     CLR.S0=3 ; CLEAR STATE ZERO BIT
:1605     SET.S1&CLR.S0=2 ; SET STATE ONE BIT AND CLEAR STATE ZERO BIT
:1606     CLR.S1&SET.S0=1 ; CLEAR STATE ONE AND SET STATE ZERO
:1607     CLR=0 ; CLEAR STATE ONE AND STATE ZERO BITS.
:1608
:1609
:1610 MISC.2/=<11:9>,.VALIDITY=<MISC.VAL>
:1611
:1612     NOP=0 ; NO OPERATION
:1613     MASK.INTERRUPTS=1 ; MASK ALL INTERRUPTS (FOR DECODE NEXT CYCLE).
:1614     ASSERT.DA.AND.PASS.WR=2 ; ASSERT DATA AVAILABLE AND PASS WR[0] TO THE ACCELERATOR OR PORT.
:1615     TRAP.ACC=3 ; TRAP ACCELERATOR
:1616     READ.ACC.UPC=4 ; READ ACCELERATOR PROGRAM COUNTER
:1617     TRANSFER.GRANT=5 ; RECOGNIZE PORT REQUEST.
:1618     MASK.HALT.AND.T.BIT=6 ; MASK HALT AND T-BIT TRAPS (FOR DECODE TWO CYCLES LATER)
:1619     WRITE.WR.TO.CONSOLE.REGISTER=7 ; SEND BITS <7:0> TO 8085.
:1620
:1621 MISC.WR/=<8:7>
:1622
:1623 MISC.WRL/=<8>
:1624 MISC.WRR/=<7>
:1625
:1626 PORT.SELECT/=<17:15>
:1627
:1628     IDC=7
  
```


:1629
:1630 PORT.FUNC/= <14:13>
:1631
:1632 READ=0
:1633 WRITE=1
:1634 CONTROL=2
:1635
:1636 R.W.FUNC/= <12:10>
:1637
:1638 CSR=0
:1639 DAR=1
:1640 DATA.BYTE=2
:1641 DATA.WORD=3
:1642 PATTERN=4
:1643 POSITION=5
:1644
:1645 CNTL.FUNC/= <12:10>
:1646
:1647 CLEAR.FIFO.CNTR=0
:1648 RESET.BR=1
:1649 CLEAR.IDC=3
:1650 SET.AUTO=4
:1651 CLEAR.AUTO=5
:1652 SEL.FIFO.A=6
:1653 SEL.FIFO.B=7
:1654

```

:1655 .PAGE
:1656 .MEMORY REQUEST FIELD
:1657
:1658 THIS FIELD IS USED TO INDICATE THE DESIRED OPERATION TO THE MEMORY CONTROL.
:1659 NOTE THAT THIS IS A DUMMY FIELD. IT IS ACTUALLY MADE UP OF TWO FIELDS.
:1660 M.FUNC1 AND M.FUNC2
:1661
:1662 THE FOLLOWING SPECIAL CONSIDERATIONS MUST BE MADE:
:1663
:1664 ROTATE.BYTE.RIGHT - PERFORMS 9-BIT ROTATE AND THEREFORE THE
:1665 ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:1666
:1667 WORD.SWAP - PERFORMS A 15-BIT ROTATE AND THERE FORE
:1668 THE ANSWER MUST BE CLEANED UP WITH A ROL WR[ ].
:1669
:1670 OCTA.WRITE.P - MUST BE LONGWORD ALLIGNED.
:1671
:1672 OCTA.WR.V.WCHK - MUST BE LONGWORD ALLIGNED. THIS DATA CAN
:1673 CROSS PAGE BOUNDARIES
:1674
:1675 MEM.FUNC/= <7> , .VALIDITY=0
:1676
:1677 :
:1678 : PREFETCH=0 : USED TO TAKE VAR AND ADD ONE BEFORE REFERENCE.
:1679 : WRITE.TB=01 : WRITE TRANSLATION BUFFER
:1680 : TEST.V.RCHK=2 : TEST WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1681 : READ.P=3 : READ WITH PHYSICAL ADDRESS
:1682 : WRITE.P=4 : WRITE WITH PHYSICAL ADDRESS
:1683 : TEST.V.WCHK=5 : WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1684 : ROTATE.BYTE.RIGHT=6 : ROTATE ADDRESS DATA 9 BITS RIGHT AND RETURN
:1685 : WORD.SWAP=7 : ROTATE ADDRESS DATA 15 BITS LEFT AND RETURN
:1686 : READ.V.NOCHK=8 : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1687 : READ.V.WCHK=9 : READ WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1688 : READ.V.RCHK=0A : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
:1689 : READ.MAINT.VAR.INC=0B : TEST VAR INCREMENTING.
:1690 : WRITE.V.NOCHK=0C : WRITE WITH VIRTUAL ADDRESS AND NO ACCESS CHECK
:1691 : WRITE.V.WCHK=0D : WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
:1692 : IB.FILL=0E : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1693 : READ.V.RCHK.IFILL=0F : READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
:1694 : WRITE.UBS.MAP=0F : WRITE TO THE UNIBUS MAP.
:1695
:1696 : OCTA.WRITE.P=10 : WRITE OCTA DATA WITH PHYSICAL ADDRESS.
:1697 : WRITE.TB.STEP=11 : WRITE TO TWO TB ENTRIES(WITH CORRECT ADDRESS)
:1698 : READ.UBS.MAP=12 : READ UNIBUS MAP
:1699 : READ.TB=13 : READ TRANSLATION BUFFER
:1700 : READ.MAINT.ADRS=14 : RPUTS VA ON UNIBUS LINES AND READS THIS DATA
:1701 : MAINT.ECC.DATA=15 : TEST ALL ECC FUNCTIONS.
:1702 : READ.CSR=16 : READ CONTROL REGISTER
:1703 : READ.CNTRL.REG=16 : READ CONTROL REGISTER
:1704 : WRITE.CNTRL.REG=17 : WRITE CONTROL REGISTER
:1705 : WRITE.CSR=17 : WRITE CONTROL REGISTER
:1706 : OCTA.READ.P=18 : READ OCTA DATA WITH PHYSICAL ADDRESS.
:1707 : READ.V.WCHK.LOCKED=19 : READ VIRTUAL ADDRESS AND WRITE ACCESS CHECK AND LOCKED
:1708 : OCTA.READ.V.RCHK=1A : READ OCTA WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
:1709 : READ.MAINT.BR.CHECK=1B : TESTS BRANCHING FUNCTION.
:1709 : ISSUE.BG=1C : ISSUES BUS GRANT (4+ADDRESS<1:0>)

```

```

:1710 OCTA.WR.V.WCHK=1D : WRITE OCTA WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK.
:1711 QUAD.READ.V.RCHK=1E : READ QUAD WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
:1712 READ.MAINT.UBS=1F : PUTS VA ON UNIBUS LINES AND READS THIS AS DATA.
:1713
:1714
:1715 M.FUNC2/=<18:16>,.VALIDITY=<DT.VAL>
:1716
:1717 M.FUNC1/=<8:7>,.VALIDITY=<DT.VAL>
:1718
:1719 PAR/=<23>,.DEFAULT=<.PARITY[<OPC2/>,<OS/>,<JUMP.ADRS/>,<JCTL/>]>
:1720 .UCODE
  
```



```

:1721 .PAGE 'MACRO DEFINITIONS VER 00.02"
:1722
:1723 THE FORMAT FOR THIS GROUP IS TWO ADDRESS WITH A MODIFY DESTINATION.
:1724 THE FIRST OPERAND IS COMBINED WITH THE SECOND OPERAND AND THE
:1725 RESULT WRITTEN TO THE SECOND OPERAND. CMF AND BIT INSTRUCTIONS ARE
:1726 READ ONLY.
:1727
:1728 DURING ALL ARITHMETIC AND LOGICAL INSTRUCTIONS THE ALU CC'S
:1729 CAN BE LOADED BY ADDING THE
:1730
:1731 DT(SIZE)&SET.ALU.CC
:1732
:1733 MACRO TO THE MICROWORD. THIS SPECIFIES THAT ALU CC'S ARE LOADED
:1734 WITH THE 2901 CONDITIONS. NO EXTERNAL MUNGING IS DONE. IF THE
:1735 DATA OPERATION IS WITH BYTES THEN THE CC'S ARE SET
:1736 ACCORDING TO BITS 7-0. WORD USES BITS 15-0 AND LONGWORD USES BITS
:1737 31-0. THE FOLLOWING IS A DESCRIPTION OF THE ALU CC'S VALUE
:1738 FOR EACH FUNCTION:
:1739
:1740 ADD -- BINARY ADDITION OF THE TWO OPERANDS
:1741
:1742 N - SIGN BIT
:1743 Z - SET IF ANSWER IS ZERO.
:1744 V - ARITHMETIC OVERFLOW
:1745 C - CARRY
:1746
:1747 SUB -- BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF THE SECOND OPERAND.
:1748
:1749 N - SIGN BIT
:1750 Z - SET IF ANSWER IS ZERO.
:1751 V - ARITHMETIC OVERFLOW
:1752 C - COMPLEMENT OF THE BORROW.
:1753
:1754 BIS -- LOGICAL OR OF THE TWO OPERANDS.
:1755
:1756 N - SIGN BIT
:1757 Z - SET IF ANSWER IS ZERO
:1758 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1759 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1760
:1761 BIC -- ONE'S COMPLEMENT OF FIRST OPERAND ANDED WITH SECOND OPERAND.
:1762
:1763 N - SIGN BIT
:1764 Z - SET IF ANSWER IS ZERO
:1765 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1766 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1767
:1768 AND -- LOGICAL AND OF THE TWO OPERANDS.
:1769
:1770 N - SIGN BIT
:1771 Z - SET IF ANSWER IS ZERO
:1772 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1773 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1774
:1775 XOR -- LOGICAL EXCLUSIVE OR OF THE TWO OPERANDS.

```

ZZ-ENKCH-1.0 ;
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC
MICRO2 1M(01)
MACRO DEFINITIONS

MACRO DEFINITIONS
19-MAY-83 13:28:52

B 4
19-MAY-83

Fiche 1 Frame B4
ENKCH Micro-diagnostic for IDC module
VER 00.02

Sequence 40
Rev 01.00

Page 37

:1776 :
:1777 :
:1778 : N - SIGN BIT
:1779 : Z - SET IF ANSWER IS ZERO
:1780 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1781 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1782 :
:1783 : CMP -- PERFORM BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF SECOND OPERAND BUT DO NOT STORE.
:1784 :
:1785 : N - SIGN BIT
:1786 : Z - SET IF ANSWER IS ZERO.
:1787 : V - ARITHMETIC OVERFLOW
:1788 : C - CARRY
:1789 :
:1790 : BIT -- PERFORM LOGICAL AND OF THE TWO OPERANDS BUT DO NOT STORE.
:1791 :
:1792 : N - SIGN BIT
:1793 : Z - SET IF ANSWER IS ZERO
:1794 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1795 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:1796 :
:1797 : SWAP -- SWITCH THE TWO VALUES.
:1798 :
:1799 : N - SIGN BIT OF VALUE TO WR.
:1800 : Z - SET IF ANSWER IS ZERO OF VALUE TO WR.
:1801 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
: C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

```

:1802 .PAGE
:1803
:1804 MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE LS[D]
:1805
:1806 ADD LS[] TO WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.PLUS.WR>,07F]>,-2]>'
:1807 SUB LS[] FROM WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.FROM.WR>,07F]>,-2]>'
:1808 BIS LS[] TO WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.OR.WR>,OFF]>,-2]>'
:1809 BIC LS[] TO WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.MASK.WR>,OFF]>,-2]>'
:1810 AND LS[] TO WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.AND.WR>,07F]>,-2]>'
:1811 XOR LS[] TO WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.XOR.WR>,07F]>,-2]>'
:1812
:1813 CMP LS[] WITH WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.CMP.WR>,OFF]>,-2]>'
:1814 BIT LS[] WITH WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,OFF]>,-2]>'
:1815 SWAP LS[] WITH WR[] 'OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/SWAP.LS.WR>,OFF]>,-2]>'
:1816
:1817 THE FOLLOWING MACROS IS TO BE USED WITH ADD,SUB,
:1818 AND,XOR LS[] TO WR[]
:1819
:1820 IT WILL TAKE THE ORIGINAL CONTENTS OF WR[]
:1821 AND PUT IT IN LS[].
:1822
:1823 XCHG 'XCHG.BIT/YES'
:1824
:1825 MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE Q-REGISTER
:1826
:1827 ADD Q TO LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS>,OFF]>,-2]>'
:1828 SUB Q FROM LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS>,OFF]>,-2]>'
:1829 BIS Q TO LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.OR.LS>,OFF]>,-2]>'
:1830 AND Q TO LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.AND.LS>,OFF]>,-2]>'
:1831 XOR Q TO LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.XOR.LS>,OFF]>,-2]>'
:1832
:1833 CMP Q WITH LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>'
:1834 BIT Q WITH LS[] 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
:1835
:1836 MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE LS[D]
:1837
:1838 ADD LS[] TO Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.Q>,OFF]>,-2]>'
:1839 SUB LS[] FROM Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q>,OFF]>,-2]>'
:1840 BIS LS[] TO Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.OR.Q>,OFF]>,-2]>'
:1841 BIC LS[] TO Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.MASK.Q>,OFF]>,-2]>'
:1842 AND LS[] TO Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.AND.Q>,OFF]>,-2]>'
:1843 XOR LS[] TO Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.XOR.Q>,OFF]>,-2]>'
:1844
:1845 CMP LS[] WITH Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.CMP.Q>,OFF]>,-2]>'
:1846 BIT LS[] WITH Q 'OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.BIT.Q>,OFF]>,-2]>'
:1847
:1848 MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE WR[B]
:1849
:1850 ADD WR[] TO LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS>,OFF]>,-2]>'
:1851 SUB WR[] FROM LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS>,OFF]>,-2]>'
:1852 BIS WR[] TO LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS>,OFF]>,-2]>'
:1853 AND WR[] TO LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS>,OFF]>,-2]>'
:1854 XOR WR[] TO LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS>,OFF]>,-2]>'
:1855
:1856 CMP WR[] WITH LS[] 'OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.CMP.LS>,OFF]>,-2]>'
    
```



```

:1857 BIT WR[] WITH LS[]      ''OPC1/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,OFF]>,-2]>''
:1858 SWAP WR[] WITH LS[]     ''SWAP LS[@2] WITH WR[@1]''
:1859
:1860 MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE WR[A]
:1861
:1862 ADD WR[] TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR>,03F]>''
:1863 SUB WR[] FROM WR[]     ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR>,03F]>''
:1864 BIS WR[] TO WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR>,03F]>''
:1865 BIC WR[] TO WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR>,03F]>''
:1866 AND WR[] TO WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR>,03F]>''
:1867 XOR WR[] TO WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>''
:1868
:1869 CMP WR[] WITH WR[]    ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.CMP.WR>,03F]>''
:1870 BIT WR[] WITH WR[]    ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.BIT.WR>,03F]>''
:1871
:1872 MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE Q-REGISTER
:1873
:1874 ADD Q TO WR[]         ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.PLUS.WR>,03F]>''
:1875 SUB Q FROM WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR>,03F]>''
:1876 BIS Q TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.OR.WR>,03F]>''
:1877 AND Q TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.AND.WR>,03F]>''
:1878 XOR Q TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.XOR.WR>,03F]>''
:1879
:1880 CMP Q WITH WR[]      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.CMP.WR>,03F]>''
:1881 BIT Q WITH WR[]     ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>''
:1882
:1883 MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE WR[B]
:1884
:1885 ADD WR[] TO Q         ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.Q>,03F]>''
:1886 SUB WR[] FROM Q      ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.FROM.Q>,03F]>''
:1887 BIS WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.OR.Q>,03F]>''
:1888 BIC WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.MASK.Q>,03F]>''
:1889 AND WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.AND.Q>,03F]>''
:1890 XOR WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.Q>,03F]>''
:1891
:1892 CMP WR[] WITH Q      ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.CMP.Q>,03F]>''
:1893 BIT WR[] WITH Q      ''OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>''
    
```

```

:1894 .PAGE
:1895
:1896 THE FORMAT OF THIS GROUP IS THREE ADDRESS TYPE. THE FIRST AND SECOND
:1897 OPERANDS ARE READ AND THE RESULT IS WRITTEN INTO THE THIRD OPERAND.
:1898
:1899 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE WR[A]
:1900
:1901 ADD WR[] PLUS WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR.Q>,03F]>'
:1902 SUB WR[] FROM WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.Q>,03F]>'
:1903 BIS WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>'
:1904 BIC WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR.Q>,03F]>'
:1905 AND WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR.Q>,03F]>'
:1906 XOR WR[] WITH WR[] TO Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>'
:1907
:1908 MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE LS[D]
:1909
:1910 ADD WR[] PLUS LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS.Q>,OFF]>,-2]>'
:1911 SUB WR[] FROM LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS.Q>,OFF]>,-2]>'
:1912 BIS WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS.Q>,OFF]>,-2]>'
:1913 AND WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS.Q>,OFF]>,-2]>'
:1914 XOR WR[] WITH LS[] TO Q 'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS.Q>,OFF]>,-2]>'
:1915
:1916 MACROS FOR THE FORM OF Q-REGISTER <-- LS[D] OPERATE WR[B]
:1917
:1918 ADD LS[] PLUS WR[] TO Q 'ADD WR[@2] PLUS LS[@1] TO Q'
:1919 SUB LS[] FROM WR[] TO Q 'OPC/BASIC,B.ADRS/@2,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.Q>,OFF]>,-2]>'
:1920 BIS LS[] WITH WR[] TO Q 'BIS WR[@2] WITH LS[@1] TO Q'
:1921 AND LS[] WITH WR[] TO Q 'AND WR[@2] WITH LS[@1] TO Q'
:1922 XOR LS[] WITH WR[] TO Q 'XOR WR[@2] WITH LS[@1] TO Q'
:1923
:1924
  
```

```

:1925 .PAGE
:1926 : SPECIAL FUNCTION ADD/SUB OPERATIONS
:1927
:1928 ADD (WR[] TO WR[])+1      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR+1>,03F]>'
:1929 ADD (LS[] TO WR[])+1      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.WR+1>,OFF]>,-2]>'
:1930 ADD (WR[] TO LS[])+1      'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS+1>,OFF]>,-2]>'
:1931
:1932 ADD OS PLUS WR[] TO LS[]  'OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/WR.PLUS.OS>,3F]>,-3]>'
:1933
:1934 SUB (WR[] FROM WR[])-1     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR-1>,03F]>'
:1935 SUB (LS[] FROM WR[])-1     'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR-1>,OFF]>,-2]>'
:1936 SUB (WR[] FROM LS[])-1     'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS-1>,OFF]>,-2]>'
:1937
:1938
:1939 SUB WRB[] FROM WRA[] TO WRB[] 'OPC1/EXTENDED,B.ADRS/@1,A.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.WR>,03F]>'
:1940 SUB LS[] FROM WR[] TO LS      'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.LS>,OFF]>,-2]>'
:1941 SUB WR[] FROM LS[] TO WR      'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WRA.FROM.LS.WRB>,OFF]>,-2]>'
:1942
:1943 SUB WRA[] FROM Q TO WRB[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.Q.WR>,03F]>'
:1944 SUB LS[] FROM Q TO LS[]       'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q.LS>,OFF]>,-2]>'
:1945 SUB Q FROM WR[] TO Q          'OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR.Q>,03F]>'
:1946 SUB Q FROM LS[] TO Q          'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS.Q>,OFF]>,-2]>'
:1947
:1948 ADD Q PLUS WR[] TO LS[]       'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.WR.TO.LS>,OFF]>,-2]>'
:1949 SUB Q FROM WR[] TO LS[]       'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.WR.TO.LS>,OFF]>,-2]>'
:1950 ADD Q PLUS LS[] TO WR[]       'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS.TO.WR>,OFF]>,-2]>'
:1951
:1952 ADD Q TO WR[] XCHG TO LS[]    'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.Q.XCHG>,OFF]>,-2]>'
:1953
    
```



```

1954 .PAGE
1955 .MOVE MACRO INSTRUCTIONS
1956
1957     DURING THE MOVE INSTRUCTIONS THE ALU CC'S CAN BE LOADED BY
1958     ADDING THE
1959
1960         DT(SIZE)&SET.ALU.CC
1961
1962     MACRO TO THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE ALU
1963     CC'S VALUE FOR EACH FUNCTION:
1964
1965     MOV -- PUT FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1966
1967     N - SIGN BIT
1968     Z - SET IF ANSWER IS ZERO
1969     V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1970     C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1971
1972     MCOM -- PUT ONE'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1973
1974     N - SIGN BIT
1975     Z - SET IF ANSWER IS ZERO
1976     V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1977     C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
1978
1979     MNEG -- PUT TWO'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
1980
1981     N - SIGN BIT
1982     Z - SET IF ANSWER IS ZERO.
1983     V - ARITHMETIC OVERFLOW
1984     C - CARRY
1985     MOV Q TO LS[]      'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.LS>,OFF]>,-2]>'
1986     MOV Q TO WR[]      'OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/MOV.Q.WR>,03F]>'
1987     MOV LS[] TO Q      'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.LS.Q>,OFF]>,-2]>'
1988     MOV Q TO WR[] XCHG TO LS[] 'OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.WR&WR.LS>,OFF]>,-2]>'
1989
1990     MOV IB.DATA TO OS   'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/IB.REQ,JCTL/NO.JUMP,TST,LD.OS/LOAD.OS'
1991     MOV CM.IB.DATA TO OS 'OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/NOP,JCTL/NO.JUMP,TST,LD.OS/LOAD.OS'
1992
1993     MOV MEM.DATA TO WR[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/WR.BACKUP'
1994     MOV MEM.DATA TO WR[] XCHG TO LS[] 'OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/@2'
1995     MOV MEM.DATA TO LS[] 'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.LS>,3F]>,-3]>'
1996     MOV LS[] TO WR[]    'OPC1/MOVE,XD.ADRS/@1,B.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.WR>,3F]>,-3]>'
1997     MOV WR[] TO LS[]    'OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.WR.LS>,3F]>,-3]>'
1998     MOV WR[] TO WR[]    'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>'
1999     MOV WR[] TO Q       'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>'
2000     MOV XWR[] TO Q      'OPC1/MOVE,XB.ADRS/@1,XD.ADRS/0,MDP/<.SHIFT[<.AND[<SDP/MOV.XWR.Q>,3F]>,-3]>'
2001
2002     MOV FPA TO LS[]     'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>'
2003     MOV PORT TO LS[]   'OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>,CC/DT(LONG)&HOLD.ALU.CC'
2004
2005     MCOM WR[] TO LS[]   'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.COM.LS>,OFF]>,-2]>'
2006     MCOM LS[] TO WR[]   'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.COM.WR>,OFF]>,-2]>'
2007     MNEG WR[] TO LS[]   'OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.NEG.LS>,OFF]>,-2]>'
2008     MNEG LS[] TO WR[]   'OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.NEG.WR>,OFF]>,-2]>'
    
```

```

:2009 MNEG WR[] TO WR[]      'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>'
:2010 MCOM WR[] TO WR[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.COM.WR>,03F]>'
:2011 MINC WR[] TO WR[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.INC.WR>,03F]>'
:2012 MDEC WR[] TO WR[]     'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>'
:2013 MNEG Q TO LSC[]       'OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.NEG.LS>,OFF]>,-2]>'
  
```

```

:2014 .PAGE
:2015
:2016 SINGLE OPERAND OPERATIONS
:2017
:2018 THIS FORM OF INSTRUCTION PERFORMS THE DESIRED OPERATION ON THE OPERAND
:2019 SPECIFIED.
:2020
:2021     CLEAR
:2022     NEGATE
:2023     INCREMENT
:2024     DECREMENT
:2025     COMPLEMENT
:2026     TEST
:2027
:2028 DURING THE SINGLE OPERAND INSTRUCTIONS THE ALU CC'S CAN BE
:2029 LOADING USING THE
:2030
:2031     DT(SIZE)&SET.ALU.CC
:2032
:2033 MACRO IN THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE
:2034 ALU CC'S FOR EACH FUNCTION:
:2035
:2036 CLR -- STORE A ZERO IN THE OPERAND.
:2037
:2038 N - SIGN BIT
:2039 Z - SET IF ANSWER IS ZERO
:2040 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2041 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2042
:2043 NEG -- PERFORM TWO'S COMPLEMENT OF THE OPERAND.
:2044
:2045 N - SIGN BIT
:2046 Z - SET IF ANSWER IS ZERO.
:2047 V - ARITHMETIC OVERFLOW
:2048 C - CARRY
:2049
:2050 INC -- ADD ONE TO THE OPERAND.
:2051
:2052 N - SIGN BIT
:2053 Z - SET IF ANSWER IS ZERO.
:2054 V - ARITHMETIC OVERFLOW
:2055 C - CARRY
:2056
:2057 DEC -- SUBTRACT ONE FROM THE OPERAND.
:2058
:2059 N - SIGN BIT
:2060 Z - SET IF ANSWER IS ZERO.
:2061 V - ARITHMETIC OVERFLOW
:2062 C - CARRY
:2063
:2064 COM -- PERFORM ONE'S COMPLEMENT OF THE OPERAND (XNOR WITH ZERO).
:2065
:2066 N - SIGN BIT
:2067 Z - SET IF ANSWER IS ZERO.
:2068 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)

```



```

:2069 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
:2070 :
:2071 : TST -- ADD ZERO TO THE OPERAND TO SET THE CONDITION CODES.
:2072 :
:2073 : N - SIGN BIT
:2074 : Z - SET IF ANSWER IS ZERO
:2075 : V - ARITHMETIC OVERFLOW (IN THIS CASE ALWAYS ZERO)
:2076 : C - CARRY (IN THIS CASE ZERO)
:2077 :
:2078 :
:2079 CLR Q "OPC1/EXTENDED,A.ADRS/0,B.ADRS/0,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>"
:2080 CLR LS[] "OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/CLR.LS>,OFF]>,-2]>"
:2081 CLR WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>"
:2082 :
:2083 NEG Q "OPC1/EXTENDED,XDP/<.AND[<SDP/NEG.Q>,03F]>"
:2084 NEG LS[] "OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/NEG.LS>,OFF]>,-2]>"
:2085 NEG WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>"
:2086 :
:2087 INC Q "OPC1/EXTENDED,XDP/<.AND[<SDP/INC.Q>,03F]>"
:2088 INC LS[] "OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/INC.LS>,OFF]>,-2]>"
:2089 INC WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.INC.WR>,03F]>"
:2090 :
:2091 DEC Q "OPC1/EXTENDED,XDP/<.AND[<SDP/DEC.Q>,03F]>"
:2092 DEC LS[] "OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/DEC.LS>,OFF]>,-2]>"
:2093 DEC WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>"
:2094 :
:2095 COM Q "OPC1/EXTENDED,XDP/<.AND[<SDP/COM.Q>,03F]>"
:2096 COM LS[] "OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/COM.LS>,OFF]>,-2]>"
:2097 COM WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.COM.WR>,03F]>"
:2098 :
:2099 TST WR[] "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>"
:2100 TST Q "OPC1/EXTENDED,XDP/<.AND[<SDP/Q.PLUS.0>,03F]>"
:2101 :
:2102 NOP "OPC/BASIC,D.ADRS/0,B.ADRS/0,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,OFF]>,-2]>"
  
```

```

2103 .PAGE
2104
2105 MACROS FOR SHIFT OPERATION ON WR[B] AND/OR Q-REGISTER
2106
2107 DURING THE SHIFT OPERATIONS THE ALU CC'S CAN BE LOADED BY
2108 USING THE
2109
2110 DT(SIZE)&SET.ALU.CC
2111
2112 MACRO WITH THE MICROWORD. THE FOLLOWING IS A DESCRIPTION OF THE
2113 ALU CC'S WITH EACH FUNCTION.
2114
2115 ROR WR[] -- ROTATE 32 BIT VALUE ONE POSITION RIGHT.
2116
2117 N - SIGN OF INPUT
2118 Z - SET IF INPUT IS ZERO
2119 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2120 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2121
2122 ROL WR[] -- ROTATE 32 BIT VALUE ONE POSITION LEFT.
2123
2124 N - SIGN OF INPUT
2125 Z - SET IF INPUT IS ZERO
2126 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2127 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2128
2129 ASHR WR[] -- SHIFT 32 BIT VALUE RIGHT ONE POSITION AND SIGN EXTEND.
2130
2131 N - SIGN OF INPUT
2132 Z - SET IF INPUT IS ZERO
2133 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2134 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2135
2136 ASHL WR[] -- SHIFT 32 BIT VALUE LEFT ONE POSITION AND INSERT A ZERO IN BOTTOM BIT.
2137
2138 N - SIGN OF INPUT
2139 Z - SET IF INPUT IS ZERO
2140 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2141 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2142
2143 ASHRC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND SIGN EXTEND.
2144
2145 N - SIGN OF INPUT WR[]
2146 Z - SET IF INPUT WR[] IS ZERO
2147 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2148 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2149
2150 RORC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION RIGHT.
2151
2152 N - SIGN OF INPUT WR[]
2153 Z - SET IF INPUT WR[] IS ZERO
2154 V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2155 C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2156
2157 ASHLC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND INSERT ZERO.
    
```

```

2158 :
2159 : N - SIGN OF INPUT WR[]
2160 : Z - SET IF INPUT WR[] IS ZERO
2161 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2162 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2163 :
2164 : ROLC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION LEFT.
2165 :
2166 : N - SIGN OF INPUT WR[]
2167 : Z - SET IF INPUT WR[] IS ZERO
2168 : V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2169 : C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
2170 :
2171 : SHL2 WR[] -- SHIFT 32 BIT VALUE TWO POSITIONS LEFT INSERTING ZEROES IN THE BOTTOM BITS.
2172 :
2173 : N - SIGN OF INPUT WR[]+WR[]
2174 : Z - SET IF INPUT WR[]+WR[] IS ZERO
2175 : V - OVERFLOW OF INPUT WR[]+WR[]
2176 : C - CARRY OF INPUT WR[]+WR[]
2177 :
2178 ROR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHR>,03F]>'
2179
2180 ROL WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHL>,03F]>'
2181
2182
2183 ASHR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHR>,03F]>'
2184
2185 ASHL WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHL>,03F]>'
2186
2187 ASHRC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
2188
2189 RORC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
2190
2191 ASHLC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>'
2192
2193 ROLC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>'
2194
2195 SHL2 WR[] 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,SI/2,XDP/<.AND[<SDP/SHL2>,03F]>'
2196
2197 COND.ADD&SHIFT WR[] TO WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>'
2198
2199 COND.SUB&SHIFT WR[] FROM WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.SUB&SHIFT>,03F]>'
2200
2201 UNSIGNED.MUL WR[] TO WR[].Q 'OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/UMUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>'
2202 SHR WR[] 'OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHR>,03F]>'
2203 SHL WR[] 'ASHL WR[@1]'
2204 SHRC WR[].Q 'OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>'
2205 SHLC WR[].Q 'ASHLC WR[].Q'
2206
2207 ; MEMORY REQUEST MACRO
2208
2209 ; * THIS MACRO IS NOT FOR GENERAL USE!!! *
2210 ; * THIS IS ONLY FOR USE IN THE FOLLOWING MACRO *
2211 MEM.FUNC[] 'M.FUNC2/<.SHIFT[<MEM.FUNC/@1>,-2]>,M.FUNC1/<.AND[<MEM.FUNC/@1>,3]>'
2212
    
```


:2213
:2214 MEM.REQ[] ADRES[] DT[] "OPC2/MEM.REQ.MEM.FUNC[]D.ADRS/@2,MDT/@3"
:2215 WRITE.MEM LS[] "OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.MEM>,3F]]>,-3]>"

```

:2216 .PAGE
:2217
:2218 .MACROS FOR MISCELLANEOUS OPERATIONS
:2219
:2220     IN THE NEBULA MICROPROGRAMMING LANGUAGE THERE IS A NEED FOR
:2221     VARIOUS UNIQUE FUNCTIONAL OPERATIONS. THESE ARE FOUND IN THE
:2222     MISC INSTRUCTIONS. MOST FUNCTIONS ARE EXECUTED BY INSERTING
:2223     THE REQUEST INSIDE THE MISC[] AND MISC2[] MACROS.
:2224
:2225 MISC []      'OPC2/MISC,MISC.1/@1,MISC.2/NOP,MISC.PORT/MISC'
:2226 MISC2 []    'OPC2/MISC,MISC.1/NOP,MISC.2/@1,MISC.PORT/MISC'
:2227 MISC [] WR[] 'MISC [a1],MISC.WRL/0,MISC.WRR/@2'
:2228 MISC2 [] WR[] 'MISC2 [a1],MISC.WRL/0,MISC.WRR/@2'
:2229
:2230 PORT.CONTROL [] 'OPC2/MISC,MISC.PORT/PORT,CNTL.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/CONTROL'
:2231 PORT.WRITE PORT.ADRS[] WITH WR[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/WRITE,MISC.WRL/0,MIS
:2232 PORT.READ PORT.ADRS[] 'OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/READ'
:2233
:2234     A FEW CASES OF THE MISCELLANEOUS FUNCTIONS HAVE BEEN CALLED OUT AS
:2235     UNIQUE FUNCTIONS IN THE MICROPROGRAMMING LANGUAGE SET. THESE
:2236     ARE LISTED BELOW.
:2237
:2238     THE FOLLOWING TO MACROS SEND DATA TO EXTERNAL ACCELERATORS.
:2239
:2240 ASSERT.DA.AND.PASS.WRO 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[0]'
:2241 ASSERT.DA.AND.PASS.WR1 'MISC2 [ASSERT.DA.AND.PASS.WR] WR[1]'
:2242
:2243     STATE REGISTER MACROS - THE POSSIBLE CHANGES TO THE STATE
:2244     BITS ARE SPECIFIED AS INDIVIDUAL FUNCTIONS.
:2245
:2246 CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR.S0,MISC.2/NOP,MISC.PORT/MISC'
:2247 CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1,MISC.2/NOP,MISC.PORT/MISC'
:2248 SET.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S0,MISC.2/NOP,MISC.PORT/MISC'
:2249 SET.STATE.ONE 'OPC2/MISC,MISC.1/SET.S1,MISC.2/NOP,MISC.PORT/MISC'
:2250 SET.STATE.ZERO&CLR.STATE.ONE 'OPC2/MISC,MISC.1/CLR.S1&SET.S0,MISC.2/NOP,MISC.PORT/MISC'
:2251 SET.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/SET.S1&CLR.S0,MISC.2/NOP,MISC.PORT/MISC'
:2252 CLR.STATE.ONE&CLR.STATE.ZERO 'OPC2/MISC,MISC.1/CLR,MISC.2/NOP,MISC.PORT/MISC'
:2253 SET.BACKUP.MASK.VALID 'OPC2/MISC,MISC.1/SET.BACKUP.MASK.VALID,MISC.2/NOP,MISC.PORT/MISC'
:2254
:2255
:2256 ; CONTEXT SIZE AND CONDITION CODE MACROS
:2257
:2258 DT(SIZE)&MAP.ALU.CC.TO.PSL 'CC/DT(SIZE)&MAP.ALU.CC.TO.PSL'
:2259 DT(LONG)&SET.ALU.CC 'CC/DT(LONG)&SET.ALU.CC'
:2260 DT(SIZE)&SET.ALU.CC 'CC/DT(SIZE)&SET.ALU.CC'
:2261
  
```

```

:2262 PAGE
:2263 JUMP AND MICRO SEQUENCER CONTROL MACROS
:2264
:2265 THE FOLLOWING MACROS ARE USED TO CONTROL THE PROGRAMS FLOW OF
:2266 CONTROL.
:2267
:2268 JMP [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY.
:2269
:2270 JMP.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF ADDRESS
:2271 IN THE INSTRUCTION AND THE OS REGISTER UNCONDITIONALLY.
:2272
:2273 JMP.IF[] TO [] -- TRANSFER TO NEW ADDRESS ONLY IF JUMP
:2274 CONDITION IS MET.
:2275
:2276 JSR [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY AND PUT
:2277 RETURN ADDRESS ON THE SEQUENCER STACK.
:2278
:2279 JSR.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF
:2280 ADDRESS IN THE INSTRUCTION AND THE OS REGISTER.
:2281 THE RETURN ADDRESS IS ALSO PUSHED ON THE SEQUENCER STACK.
:2282
:2283 RETURN -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER STACK AND
:2284 POP THE STACK.
:2285
:2286 RETURN+1 -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:2287 STACK PLUS ONE AND POP THE STACK.
:2288
:2289 RETURN+1.IF(MEM.REF.OK) -- IF THE MOST RECENT MEMORY REQUEST HAD
:2290 NO ERROR THEN TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
:2291 STACK PLUS ONE AND POP THE STACK. IF THERE WAS AN
:2292 ERROR THEN SKIP OVER THE ENEXT INSTRUCTION.
:2293
:2294 LOOP.IF(NEQ) -- IF THE ALU Z-BIT IS ZERO (LAST VALUE CALCULATED
:2295 WHEN THE ALU.CC'S WERE SET DID NOT RESULT AS ZERO)
:2296 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:2297 STACK. THE STACK IS NOT POPPED. IF THE ALU Z-BIT
:2298 IS ONE THEN CONTINUE WITH THE NEXT INSTRUCTION.
:2299 (UPC+1) AND THE STACK IS POPPED.
:2300
:2301 LOOP.IF(GEQU) -- IF THE ALU C-BIT IS ONE (LAST VALUE CALCULATED WHEN
:2302 THE ALU.CC'S WERE SET WAS GREATER THAN OR EQUAL TO ZERO)
:2303 THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
:2304 STACK. THE STACK IS NOT POPPED. IF THE ALU C-BIT
:2305 IS ZERO THEN CONTINUE WITH THE NEXT INSTRUCTION
:2306 (UPC+1) AND THE STACK IS POPPED.
:2307
:2308 LOOP.IF(NO INTERRUPT AND NO SYNC) -- IF THE INTERRUPT CONDITION
:2309 IS NOT TRUE AND THE ACCELERATOR SYNC CONDITION IS NOT
:2310 TRUE THEN TRANSFER TO THE ADDRESS ON THE TOP OF THE SEQUENCER
:2311 STACK. THE STACK IS NOT POPPED. IF THE INTERRUPT
:2312 CONDITION IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:2313 (UPC+1) AND THE SEQUENCER STACK IS NOT POPPED. IF THE ACCELERATOR
:2314 SYNC IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
:2315 (UPC+1) AND THE SEQUENCER STACK IS POPPED.
:2316
    
```



```

:2317 :      SKIP.IF[] -- IF THE SKIP CONDITION IS SATISFIED THEN TRANSFER
:2318 :      TO UPC+2 ELSE TRANSFER TO UPC+1.
:2319 :
:2320 :
:2321 JMP []      "OPC2/JUMP,JUMP.ADRS/@1,JCTL/JUMP"
:2322 JMP.OS []   "OPC2/JUMP,JUMP.ADRS/@1,OS/WITH.OS,JCTL/JUMP"
:2323 JMP.IF[] TO [] "OPC2/JUMP,JCTL/@1,JUMP.ADRS/@2"
:2324 JSR []     "OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1"
:2325 JSR.OS []  "OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1,OS/WITH.OS"
:2326 :
:2327 RETURN     "SCTL/RETURN"
:2328 RETURN+1   "SCTL/RETURN+1"
:2329 RETURN+1.IF(MEM.REF.OK) "SCTL/RET.NOERR"
:2330 :
:2331 LOOP.IF(NEQ) "SCTL/JMP.Z.CLR"
:2332 LOOP.IF(GEQU) "SCTL/JMP.C.SET"
:2333 LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC) "SCTL/LOOP.IF.NO.I.AND.SYNC"
:2334 :
:2335 SKIP.IF[]    "SCTL/@1"
:2336 SKIP        "SCTL/SKIP"
  
```

```

:2337 .PAGE
:2338
:2339 : INSTRUCTION STREAM MACROS
:2340
:2341 * THIS MACRO IS NOT FOR GENERAL USE!!!
:2342 * THIS IS ONLY FOR USE IN THE FOLLOWING MACROS
:2343 DEC[] "OPC2/DECODE,DEC.ADRS/<.SHIFT[<JUMP.ADRS/@1>,-8]>"
:2344
:2345 DECODE.SPEC ADRS[] "DEC[@1],IFUNC/SPEC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/S
:2346 DECODE.ASRC ADRS[] "DEC[@1],IFUNC/ASRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/A
:2347 DECODE.ESRC ADRS[] "DEC[@1],IFUNC/ESRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/E
:2348 DECODE.VSRC ADRS[] "DEC[@1],IFUNC/VSRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/V
:2349 DECODE.FLOAT ADRS[] "DEC[@1],IFUNC/FLOAT,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/
:2350 DECODE.OPC1 ADRS[] "DEC[@1],IFUNC/VAX.IRD,R.DST/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD"
:2351 EXEC.DISPATCH ADRS[] "DEC[@1],IFUNC/VAX.EXEC,IB.REQ/NOP,R.DST/NOP,CM.HI/LOW.BYTE,JCTL/JSR,OPC.SPEC/VAX.EXEC"
:2352
:2353 DECODE.CM.OPC ADRS[] "DEC[@1],IFUNC/CM.IRD,CM.HI/HI.BYTE,OPC.SPEC/CM.IRD,LD.OS/LOAD.OS"
:2354 DECODE.CM.DST ADRS[] "DEC[@1],IFUNC/CM.DST,OPC.SPEC/CM.DST,LD.OS/LOAD.OS,R.DST/ENAB"
:2355 DECODE.CM.SINGLE ADRS[] "DEC[@1],IFUNC/CM.SINGLE,OPC.SPEC/CM.SINGLE"
:2356 CM.EXEC ADRS[] "DEC[@1],IFUNC/CM.EXEC,JCTL/JUMP,OPC.SPEC/CM.EXEC,LD.OS/LOAD.OS"
:2357
:2358 SAVE.PC WRO "BPC/SAVE.PC"
:2359 SAVE.PC WR4 "BPC/SAVE.PC"
:2360 SAVE.CM.PC WRO "BPC/SAVE.PC"
:2361 SAVE.CM.PC WR4 "BPC/SAVE.PC"
:2362
:2363 : THESE SKIP CONDITIONS ARE USED FOR THE DECODE MICRO INSTRUCTION
:2364
:2365 SKIP.IF(1B VALID) "JCTL/NO.JUMP.TST"
:2366 JMP.IF(1B VALID) "JCTL/JUMP.VALID"
:2367 JSR.IF(1B VALID) "JCTL/JSR.VALID"
:2368
:2369 :
:2370 THESE SKIP CONDITIONS ARE TO BE USED WITH THE COMPATIBILITY MODE DECODE
:2371 MACROS.
:2372
:2373 JMP.TO [] "JCTL/JUMP,DEC[@1]"
:2374 JSR.TO [] "JCTL/JSR,DEC[@1]"
:2375 .BIN
    
```

```

:2376 .PAGE
:2377 .TOC "FIELD DEFINITIONS FOR DATA PATH CONTROL MICRO WORD VER 00.01"
:2378
:2379 : THIS SECTION IS FOR THE DATA PATH CONTROL ROMS
:2380 :
:2381
:2382 .CCODE
:2383 SDP/=<8:0>,,ADDRESS,,VALIDITY=0
:2384
:2385 : THIS FIELD CORRESPONDS TO THE 2901 ALU FUNCTION CONTROL PINS ... 15,14,13
:2386
:2387 ALU/=<2:0>,,DEFAULT=<ALU/R.OR.S>
:2388
:2389 R.PLUS.S=0 ; ADD R WITH S
:2390 S.MINUS.R=1 ; SUBTRACT R FROM S
:2391 R.MINUS.S=2 ; SUBTRACT S FROM R
:2392 R.OR.S=3 ; OR R WITH S
:2393 R.AND.S=4 ; AND R WITH S
:2394 NOTR.AND.S=5 ; COMPLEMENT R THEN AND WITH S
:2395 R.XOR.S=6 ; XOR R WITH S
:2396 R.XNOR.S=7 ; XOR R WITH S THEN COMPLEMENT THE RESULT
:2397 SRC/=<8:6>,,DEFAULT=<SRC/D.0>
:2398
:2399 O.Q=2 ; RMX=0 SMX=Q
:2400 O.B=3 ; RMX=0 SMX=B
:2401 A.Q=0 ; RMX=A SMX=Q
:2402 A.B=01 ; RMX=A SMX=B
:2403 D.Q=06 ; RMX=D SMX=Q
:2404 D.O=7 ; RMX=D SMX=0
:2405 O.A=4 ; RMX=0 SMX=A
:2406 D.A=5 ; RMX=D SMX=A
:2407
:2408 : THIS FIELD CORRESPONDES TO THE 2901 ALU DESTINATION
:2409 : CONTROL PINS 18, 17 AND 16.
:2410
:2411 DST/=<5:3>,,DEFAULT=<DST/NOP>
:2412
:2413 LOAD.Q=0 ; LOAD Q-REGISTER
:2414 NOP=1 ; HOLD ALL REGISTERS
:2415 WRITE.B.A=2 ; WRITE RAM B-PORT AND OUTPUT RAM A-PORT
:2416 WRITE.B.F=3 ; WRITE RAM B-PORT AND OUTPUT ALU
:2417 RSHF.RAM.Q=4 ; RIGHT SHIFT RAM AND Q
:2418 RSHF.RAM=5 ; RIGHT SHIFT RAM
:2419 LSHF.RAM.Q=6 ; LEFT SHIFT RAM AND Q
:2420 LSHF.RAM=7 ; LEFT SHIFT RAM
:2421
:2422 CIN/=<9>,,DEFAULT=0
:2423
:2424 NO.CIN=0 ; NO CARRY IN TO ALU
:2425 CIN=1 ; CARRY IN TO ALU
:2426
:2427
:2428
  
```



```

:2429 .PAGE
:2430 .TOC "CONTROL ROM CONTENTS VER 00.01"
:2431 .SEQUENTIAL
:2432 : THESE ARE THE DATA PATH CONTROL ROM CONTENTS
:2433 :
:2434 : THE FOLLOWING LOCATIONS ARE USED BY THE DECODE.IS MICRO INSTRUCTION
:2435 :
:2436 : THIS INSTRUCTION WILL DO PC+1 OPERATIONS
:2437 :
:2438 : BACK UP PC IN 2901 RAM
:2439 000:
:2440 DECODE.IS:
C 000, 3D8 :2441 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 001, 3D8 :2442 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 002, 3D8 :2443 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 003, 3D8 :2444 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 004, 3D8 :2445 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 005, 3D8 :2446 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 006, 3D8 :2447 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 007, 3D8 :2448 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 008, 3D8 :2449 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 009, 3D8 :2450 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00A, 3D8 :2451 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00B, 3D8 :2452 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00C, 3D8 :2453 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00D, 3D8 :2454 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00E, 3D8 :2455 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 00F, 3D8 :2456 SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2457 :
:2458 :
:2459 : DON'T BACK UP PC
C 010, 3C8 :2460 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 011, 3C8 :2461 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 012, 3C8 :2462 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 013, 3C8 :2463 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 014, 3C8 :2464 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 015, 3C8 :2465 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 016, 3C8 :2466 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 017, 3C8 :2467 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 018, 3C8 :2468 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 019, 3C8 :2469 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01A, 3C8 :2470 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01B, 3C8 :2471 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01C, 3C8 :2472 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01D, 3C8 :2473 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01E, 3C8 :2474 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01F, 3C8 :2475 SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN

```

	:2476	.PAGE	
	:2477	:	THESE ADDRESSES ARE USED BY THE 'JUMP' MICRO INSTRUCTION
	:2478	:	
	:2479	:	THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
	:2480	:	THE 2901 DURING JUMP OR JSR OPERATIONS.
	:2481	:	
	:2482	:	020:
	:2483	:	JUMP:
C 020.	3CB	:2484	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 021.	3CB	:2485	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 022.	3CB	:2486	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 023.	3CB	:2487	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 024.	3CB	:2488	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 025.	3CB	:2489	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 026.	3CB	:2490	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 027.	3CB	:2491	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 028.	3CB	:2492	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 029.	3CB	:2493	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02A.	3CB	:2494	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02B.	3CB	:2495	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02C.	3CB	:2496	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02D.	3CB	:2497	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02E.	3CB	:2498	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 02F.	3CB	:2499	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 030.	3CB	:2500	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 031.	3CB	:2501	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 032.	3CB	:2502	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 033.	3CB	:2503	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 034.	3CB	:2504	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 035.	3CB	:2505	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 036.	3CB	:2506	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 037.	3CB	:2507	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 038.	3CB	:2508	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 039.	3CB	:2509	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03A.	3CB	:2510	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03B.	3CB	:2511	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03C.	3CB	:2512	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03D.	3CB	:2513	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03E.	3CB	:2514	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 03F.	3CB	:2515	SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN

:2516 .PAGE
 :2517 : THESE LOCATIONS ARE USED BY THE MISC' MICRO INSTRUCTION
 :2518 :
 :2519 : THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
 :2520 : THE 2901 DURING MISC INSTRUCTION EXECUTION.
 :2521 :

:2522 040:
 :2523 MISC:

C 040.	313	:2524	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 041.	313	:2525	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 042.	313	:2526	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 043.	313	:2527	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 044.	313	:2528	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 045.	313	:2529	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 046.	313	:2530	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 047.	313	:2531	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 048.	313	:2532	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 049.	313	:2533	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04A.	313	:2534	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04B.	313	:2535	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04C.	313	:2536	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04D.	313	:2537	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04E.	313	:2538	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 04F.	313	:2539	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 050.	313	:2540	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 051.	313	:2541	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 052.	313	:2542	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 053.	313	:2543	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 054.	313	:2544	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 055.	313	:2545	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 056.	313	:2546	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 057.	313	:2547	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 058.	313	:2548	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 059.	313	:2549	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05A.	313	:2550	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05B.	313	:2551	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05C.	313	:2552	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05D.	313	:2553	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05E.	313	:2554	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 05F.	313	:2555	SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN

	:2556	.PAGE	
	:2557	: THESE ADDRESSES ARE USED BY THE MEM.REQ MICRO INSTRUCTION	
	:2558		
	:2559	: THIS INSTRUCTION CAUSES WR[5] TO BE LOADED WITH THE VIRTUAL ADDRESS	
	:2560	: OF THE MEMORY REFERENCE.	
	:2561	060:	
	:2562	MEM.REQ:	
C 060,	3DB	:2563	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 061,	3DB	:2564	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 062,	3DB	:2565	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 063,	3DB	:2566	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 064,	3DB	:2567	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 065,	3DB	:2568	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 066,	3DB	:2569	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 067,	3DB	:2570	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 068,	3DB	:2571	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 069,	3DB	:2572	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06A,	3DB	:2573	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06B,	3DB	:2574	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06C,	3DB	:2575	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06D,	3DB	:2576	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06E,	3DB	:2577	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 06F,	3DB	:2578	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 070,	3DB	:2579	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 071,	3DB	:2580	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 072,	3DB	:2581	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 073,	3DB	:2582	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 074,	3DB	:2583	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 075,	3DB	:2584	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 076,	3DB	:2585	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 077,	3DB	:2586	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 078,	3DB	:2587	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 079,	3DB	:2588	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07A,	3DB	:2589	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07B,	3DB	:2590	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07C,	3DB	:2591	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07D,	3DB	:2592	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07E,	3DB	:2593	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 07F,	3DB	:2594	SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN

```

:2595 .PAGE
:2596 : THESE ARE THE ADDRESSES FOR THE EXTENDED MICRO INSTRUCTION
:2597 : ADDRESSES START AT 80-FF
:2598 080:
:2599
C 080, 118 :2600 WR.PLUS.O.TO.WR:
:2601 SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 081, 318 :2602 WR.INC.WR:
:2603 SRC/O.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 082, 31A :2604 WR.NEG.WR:
:2605 SRC/O.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 083, 31F :2606 WR.COM.WR:
:2607 SRC/O.A,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
:2608
C 084, 119 :2609 WR.DEC.WR:
:2610 SRC/O.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 085, 10B :2611 FILL85:
:2612 SRC/O.A
C 086, 29B :2613 MOV.Q.WR:
:2614 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 087, 08B :2615 FILL87:
:2616 SRC/O.Q
:2617
C 088, 060 :2618 COND.ADD&SHIFT:
:2619 SRC/A.B,ALU/R.PLUS.S,DST/RSHF.RAM.Q,CIN/NO.CIN
C 089, 261 :2620 COND.SUB&SHIFT:
:2621 SRC/A.B,ALU/S.MINUS.R,DST/RSHF.RAM.Q,CIN/CIN
C 08A, 04B :2622 FILL8A:
:2623 SRC/A.B
C 08B, 078 :2624 SHL2:
:2625 SRC/A.B,ALU/R.PLUS.S,DST/LSHF.RAM,CIN/NO.CIN
:2626
C 08C, 0E3 :2627 ASHRC:
:2628 SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM.Q,CIN/NO.CIN
C 08D, 2EB :2629 ASHR:
:2630 SRC/O.B,ALU/R.OR.S,DST/RSHF.RAM,CIN/CIN
C 08E, 2F3 :2631 ASHLC:
:2632 SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM.Q,CIN/CIN
C 08F, 2FB :2633 ASHL:
:2634 SRC/O.B,ALU/R.OR.S,DST/LSHF.RAM,CIN/CIN
:2635
C 090, 088 :2636 Q.PLUS.O:
:2637 SRC/O.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 091, 08B :2638 FILL91:
:2639 SRC/O.Q
C 092, 20A :2640 WR.CMP.Q:
:2641 SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 093, 209 :2642 Q.CMP.WR:
:2643 SRC/A.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
:2644
C 094, 081 :2645 DEC.Q:
:2646 SRC/O.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/NO.CIN
C 095, 280 :2647 INC.Q:
:2648 SRC/O.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/CIN
:2649 NEG.Q:

```

ZZ-ENKCH-1.0	:	ENKCH.MIC	CONTROL ROM CONTENT	K 5	19-MAY-83	Fiche 1	Frame K5	Sequence 62	
:	:	ENKCH.MCR	MICRO2 1M(01)		13:28:52	ENKCH Micro-diagnostic	for IDC module	Rev 01.00	Page 59
:	:	ENKCH.MIC	CONTROL ROM CONTENTS			VER 00.01			

C 096, 282	:2650	SRC/O.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2651	COM.Q:
C 097, 287	:2652	SRC/O.Q,ALU/R.XNOR.S,DST/LOAD.Q,CIN/CIN
	:2653	
	:2654	WR.BIT.WR:
C 098, 04C	:2655	SRC/A.B,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
	:2656	WR.CMP.WR:
C 099, 24A	:2657	SRC/A.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2658	FILL9A:
C 09A, 04B	:2659	SRC/A.B
	:2660	WR.PLUS.WR+1:
C 09B, 258	:2661	SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
	:2662	
	:2663	FILL9C:
C 09C, 04B	:2664	SRC/A.B
	:2665	FILL9D:
C 09D, 04B	:2666	SRC/A.B
	:2667	FILL9E:
C 09E, 0CB	:2668	SRC/O.B
	:2669	FILL9F:
C 09F, 0CB	:2670	SRC/O.B
	:2671	
	:2672	WR.PLUS.Q:
C 0A0, 000	:2673	SRC/A.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2674	WR.FROM.Q:
C 0A1, 201	:2675	SRC/A.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2676	Q.FROM.WR.Q:
C 0A2, 202	:2677	SRC/A.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2678	WR.OR.Q:
C 0A3, 203	:2679	SRC/A.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2680	
	:2681	WR.AND.Q:
C 0A4, 004	:2682	SRC/A.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2683	WR.MASK.Q:
C 0A5, 205	:2684	SRC/A.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
	:2685	WR.XOR.Q:
C 0A6, 206	:2686	SRC/A.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2687	FILLA7:
C 0A7, 00B	:2688	SRC/A.Q
	:2689	
	:2690	WR.PLUS.WR.Q:
C 0A8, 040	:2691	SRC/A.B,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2692	WR.FROM.WR.Q:
C 0A9, 241	:2693	SRC/A.B,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2694	FILLAA:
C 0AA, 04B	:2695	SRC/A.B
	:2696	WR.OR.WR.Q:
C 0AB, 243	:2697	SRC/A.B,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2698	
	:2699	WR.AND.WR.Q:
C 0AC, 044	:2700	SRC/A.B,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2701	WR.MASK.WR.Q:
C 0AD, 245	:2702	SRC/A.B,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
	:2703	WR.XOR.WR.Q:
C 0AE, 246	:2704	SRC/A.B,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN

ZZ-ENKCH-1.0 :
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

CONTROL ROM CONTENT 19-MAY-83

MICRO2 1M(01) 19-MAY-83 13:28:52
CONTROL ROM CONTENTS

Fiche 1 Frame L5

Sequence 63

ENKCH Micro-diagnostic for IDC module
VER 00.01

Rev 01.00

Page 60

C 0AF, 04B	:2705	FILLAF:
	:2706	SRC/A.B
	:2707	
C 0B0, 018	:2708	Q.PLUS.WR:
	:2709	SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0B1, 219	:2710	WR.FROM.Q.WR:
	:2711	SRC/A.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0B2, 21A	:2712	Q.FROM.WR:
	:2713	SRC/A.Q,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0B3, 21B	:2714	Q.OR.WR:
	:2715	SRC/A.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
	:2716	
C 0B4, 01C	:2717	Q.AND.WR:
	:2718	SRC/A.Q,ALU/R.AND.S,DST/WRITE.B.F,CIN/NO.CIN
C 0B5, 00B	:2719	FILLB5:
	:2720	SRC/A.Q
C 0B6, 21E	:2721	Q.XOR.WR:
	:2722	SRC/A.Q,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 0B7, 20C	:2723	Q.BIT.WR:
	:2724	SRC/A.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
	:2725	
C 0B8, 058	:2726	WR.PLUS.WR:
	:2727	SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0B9, 259	:2728	WR.FROM.WR:
	:2729	SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0BA, 25A	:2730	WR.FROM.WR.WR:
	:2731	SRC/A.B,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0BB, 25B	:2732	WR.OR.WR:
	:2733	SRC/A.B,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
	:2734	
C 0BC, 059	:2735	WR.FROM.WR-1:
	:2736	SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0BD, 25D	:2737	WR.MASK.WR:
	:2738	SRC/A.B,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0BE, 25E	:2739	WR.XOR.WR:
	:2740	SRC/A.B,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 0BF, 25C	:2741	WR.AND.WR:
	:2742	SRC/A.B,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
	:2743	

```

:2744 .PAGE
:2745 : THIS IS THE DP CODES FOR THE 'MOVE' MICRO INSTRUCTION
:2746
:2747 0C0:
:2748
:2749 MOV.MEM.WR:
C 0C0. 1D3 :2750 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C1. 1D3 :2751 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C2. 1D3 :2752 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C3. 1D3 :2753 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C4. 1D3 :2754 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C5. 1D3 :2755 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C6. 1D3 :2756 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0C7. 1D3 :2757 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:2758
:2759 MOV.LS.MEM:
C 0C8. 1CB :2760 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0C9. 1CB :2761 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CA. 1CB :2762 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CB. 1CB :2763 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CC. 1CB :2764 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CD. 1CB :2765 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CE. 1CB :2766 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 0CF. 1CB :2767 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2768
:2769 MOV.XWR.Q:
C 0D0. 0C3 :2770 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D1. 0C3 :2771 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D2. 0C3 :2772 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D3. 0C3 :2773 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D4. 0C3 :2774 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D5. 0C3 :2775 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D6. 0C3 :2776 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0D7. 0C3 :2777 SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
:2778
:2779 MOV.LS.WR:
C 0D8. 1DB :2780 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0D9. 1DB :2781 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DA. 1DB :2782 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DB. 1DB :2783 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DC. 1DB :2784 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DD. 1DB :2785 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DE. 1DB :2786 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 0DF. 1DB :2787 SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
:2788
  
```

```

:2789 .PAGE
:2790 MOV.MEM.LS:
C OE0, 1CB :2791 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE1, 1CB :2792 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE2, 1CB :2793 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE3, 1CB :2794 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE4, 1CB :2795 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE5, 1CB :2796 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE6, 1CB :2797 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE7, 1CB :2798 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2799 MOV.ACC.LS:
C OE8, 1CB :2800 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OE9, 1CB :2801 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OEA, 1CB :2802 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OEB, 1CB :2803 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OEC, 1CB :2804 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OED, 1CB :2805 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OEE, 1CB :2806 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OEF, 1CB :2807 SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2808
:2809
:2810 WR.PLUS.OS:
C OF0, 148 :2811 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF1, 148 :2812 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF2, 148 :2813 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF3, 148 :2814 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF4, 148 :2815 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF5, 148 :2816 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF6, 148 :2817 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C OF7, 148 :2818 SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
:2819
:2820 MOV.WR.LS:
C OF8, 10B :2821 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OF9, 10B :2822 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFA, 10B :2823 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFB, 10B :2824 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFC, 10B :2825 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFD, 10B :2826 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFE, 10B :2827 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C OFF, 10B :2828 SRC/O.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
:2829
    
```


:2830 .PAGE
:2831 : THIS GROUP OF INSTRUCTIONS IS FOR THE 'BASIC' MICRO INSTRUCTION
:2832 : THIS USES ADDRESSES 100-1FF
:2833
:2834 100:
:2835 LS.PLUS.WR:
C 100, 158 :2836 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 101, 158 :2837 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 102, 158 :2838 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 103, 158 :2839 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
:2840 LS.FROM.WR:
C 104, 359 :2841 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 105, 359 :2842 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 106, 359 :2843 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 107, 359 :2844 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
:2845 LS.AND.WR:
C 108, 35C :2846 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 109, 35C :2847 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 10A, 35C :2848 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 10B, 35C :2849 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
:2850 LS.XOR.WR:
C 10C, 35E :2851 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 10D, 35E :2852 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 10E, 35E :2853 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 10F, 35E :2854 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
:2855
:2856 LS.FROM.WR-1:
C 110, 159 :2857 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 111, 159 :2858 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 112, 159 :2859 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 113, 159 :2860 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
:2861 LS.MASK.WR:
C 114, 35D :2862 SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 115, 35D :2863 SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 116, 35D :2864 SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 117, 35D :2865 SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
:2866 LS.PLUS.WR+1:
C 118, 358 :2867 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 119, 358 :2868 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 11A, 358 :2869 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 11B, 358 :2870 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
:2871 LS.OR.WR:
C 11C, 35B :2872 SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 11D, 35B :2873 SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 11E, 35B :2874 SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 11F, 35B :2875 SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
:2876
:2877 WR.PLUS.LS.Q:
C 120, 140 :2878 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 121, 140 :2879 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 122, 140 :2880 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 123, 140 :2881 SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
:2882 LS.FROM.WR.Q:
C 124, 341 :2883 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 125, 341 :2884 SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN

ZZ-ENKCH-1.0	:	ENKCH.MIC	CONTROL ROM CONTENT	19-MAY-83	Fiche 1 Frame C6	Sequence 67	Page 64
: ENKCH.MCR	:	MICRO2 1M(01)	19-MAY-83 13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00		
: ENKCH.MIC	:	CONTROL ROM CONTENTS		VER 00.01			

C 126, 341	:2885	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 127, 341	:2886	SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2887	WR.FROM.LS.Q:
C 128, 342	:2888	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 129, 342	:2889	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 12A, 342	:2890	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 12B, 342	:2891	SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2892	WR.OR.LS.Q:
C 12C, 343	:2893	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 12D, 343	:2894	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 12E, 343	:2895	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 12F, 343	:2896	SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2897	
	:2898	WR.AND.LS.Q:
C 130, 144	:2899	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 131, 144	:2900	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 132, 144	:2901	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 133, 144	:2902	SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2903	WR.XOR.LS.Q:
C 134, 346	:2904	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 135, 346	:2905	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 136, 346	:2906	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 137, 346	:2907	SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2908	LS.CMP.WR:
C 138, 34A	:2909	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 139, 34A	:2910	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 13A, 34A	:2911	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 13B, 34A	:2912	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2913	WR.CMP.LS:
C 13C, 349	:2914	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 13D, 349	:2915	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 13E, 349	:2916	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 13F, 349	:2917	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2918	
	:2919	LS.PLUS.Q:
C 140, 180	:2920	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 141, 180	:2921	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 142, 180	:2922	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 143, 180	:2923	SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
	:2924	LS.FROM.Q:
C 144, 381	:2925	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 145, 381	:2926	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 146, 381	:2927	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 147, 381	:2928	SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
	:2929	Q.FROM.LS.Q:
C 148, 382	:2930	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 149, 382	:2931	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 14A, 382	:2932	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 14B, 382	:2933	SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
	:2934	LS.OR.Q:
C 14C, 383	:2935	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 14D, 383	:2936	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 14E, 383	:2937	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 14F, 383	:2938	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
	:2939	

C 150, 185	:2940	LS.MASK.Q:	SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 151, 185	:2941		SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 152, 185	:2942		SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 153, 185	:2943		SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
	:2944	LS.XOR.Q:	SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 154, 386	:2945		SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 155, 386	:2946		SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 156, 386	:2947		SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 157, 386	:2948		SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
	:2949	LS.AND.Q:	SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 158, 384	:2950		SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 159, 384	:2951		SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 15A, 384	:2952		SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 15B, 384	:2953		SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
	:2954	LS.BIT.Q:	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 15C, 38C	:2955		SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 15D, 38C	:2956		SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 15E, 38C	:2957		SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 15F, 38C	:2958		SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
	:2959	MOV.LS.Q:	SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 160, 1C3	:2960		SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 161, 1C3	:2961		SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 162, 1C3	:2962		SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 163, 1C3	:2963		SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
	:2964	WR.BIT.LS:	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 164, 34C	:2965		SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 165, 34C	:2966		SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 166, 34C	:2967		SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 167, 34C	:2968		SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
	:2969	LS.CMP.Q:	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 168, 38A	:2970		SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 169, 38A	:2971		SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 16A, 38A	:2972		SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 16B, 38A	:2973		SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:2974	Q.CMP.LS:	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 16C, 389	:2975		SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 16D, 389	:2976		SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 16E, 389	:2977		SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 16F, 389	:2978		SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:2979	Q.PLUS.LS.TO.WR:	SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 170, 198	:2980		SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 171, 198	:2981		SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 172, 198	:2982		SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 173, 198	:2983		SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
	:2984	WRA.FROM.LS.WRB:	SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 174, 35A	:2985		SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 175, 35A	:2986		SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 176, 35A	:2987		SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 177, 35A	:2988		SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
	:2989	LS.NEG.WR:	SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 178, 3D9	:2990		SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 179, 3D9	:2991		SRC/D.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
	:2992		
	:2993		
	:2994		

ZZ-ENKCH-1.0
: ENKCH.MCR
: ENKCH.MIC

: ENKCH.MIC
MICRO2 1M(01)
CONTROL ROM CONTENTS

CONTROL ROM CONTENTS
19-MAY-83 13:28:52
ENKCH Micro-diagnostic for IDC module
VER 00.01

Fiche 1 Frame E6
Sequence 69
Page 66

C 17A, 3D9	:2995	SRC/D.O,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 17B, 3D9	:2996	SRC/D.O,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
	:2997	LS.COM.WR:
C 17C, 3DF	:2998	SRC/D.O,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 17D, 3DF	:2999	SRC/D.O,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 17E, 3DF	:3000	SRC/D.O,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 17F, 3DF	:3001	SRC/D.O,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN

```

:3002 .PAGE
:3003
:3004 THE FOLLOWING LOCATIONS HAVE THE LOCAL STORE AS THEIR DESTINATION.
:3005
:3006 THIS FACT IS USED BY HARDWARE TO GENERATE LOCAL STORE WRITE PULSES
:3007
:3008 180:
:3009
:3010 LS.PLUS.WR.XCHG:
C 180, 150 :3011 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 181, 150 :3012 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 182, 150 :3013 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 183, 150 :3014 SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:3015 LS.FROM.WR.XCHG:
C 184, 351 :3016 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 185, 351 :3017 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 186, 351 :3018 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 187, 351 :3019 SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
:3020 LS.AND.WR.XCHG:
C 188, 354 :3021 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 189, 354 :3022 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 18A, 354 :3023 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 18B, 354 :3024 SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
:3025 LS.XOR.WR.XCHG:
C 18C, 356 :3026 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 18D, 356 :3027 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 18E, 356 :3028 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 18F, 356 :3029 SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
:3030
:3031 SWAP.LS.WR:
C 190, 1D3 :3032 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 191, 1D3 :3033 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 192, 1D3 :3034 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 193, 1D3 :3035 SRC/D.O,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
:3036 CLR.LS:
C 194, 3CC :3037 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 195, 3CC :3038 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 196, 3CC :3039 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
C 197, 3CC :3040 SRC/D.O,ALU/R.AND.S,DST/NOP,CIN/CIN
:3041 MOV.Q.WR&WR.LS:
C 198, 293 :3042 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 199, 293 :3043 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 19A, 293 :3044 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 19B, 293 :3045 SRC/O.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
:3046 WR.PLUS.Q.XCHG:
C 19C, 010 :3047 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 19D, 010 :3048 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 19E, 010 :3049 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 19F, 010 :3050 SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
:3051
:3052 WR.FROM.LS-1:
C 1A0, 14A :3053 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1A1, 14A :3054 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1A2, 14A :3055 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1A3, 14A :3056 SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
    
```

ZZ-ENKCH-1.0 :
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC
MICRO2 1M(01)
CONTROL ROM CONTENTS

6 6
CONTROL ROM CONTENT19-MAY-83
19-MAY-83 13:28:52

Fiche 1 Frame G6
ENKCH Micro-diagnostic for IDC module
VER 00.01

Sequence 71
Rev 01.00 Page 68

C 1A4, 349	:3057	LS.FROM.WR.LS:
C 1A5, 349	:3058	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1A6, 349	:3059	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1A7, 349	:3060	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:3061	SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:3062	WR.PLUS.LS+1:
C 1A8, 348	:3063	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1A9, 348	:3064	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1AA, 348	:3065	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1AB, 348	:3066	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
	:3067	WR.FROM.LS:
C 1AC, 34A	:3068	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1AD, 34A	:3069	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1AE, 34A	:3070	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1AF, 34A	:3071	SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3072	
	:3073	WR.PLUS.LS:
C 1B0, 148	:3074	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1B1, 148	:3075	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1B2, 148	:3076	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1B3, 148	:3077	SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:3078	WR.OR.LS:
C 1B4, 34B	:3079	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1B5, 34B	:3080	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1B6, 34B	:3081	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1B7, 34B	:3082	SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
	:3083	WR.AND.LS:
C 1B8, 34C	:3084	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 1B9, 34C	:3085	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 1BA, 34C	:3086	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 1BB, 34C	:3087	SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
	:3088	WR.XOR.LS:
C 1BC, 34E	:3089	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1BD, 34E	:3090	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1BE, 34E	:3091	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1BF, 34E	:3092	SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
	:3093	
	:3094	Q.PLUS.LS:
C 1C0, 188	:3095	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1C1, 188	:3096	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1C2, 188	:3097	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1C3, 188	:3098	SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:3099	LS.FROM.Q.LS:
C 1C4, 389	:3100	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1C5, 389	:3101	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1C6, 389	:3102	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1C7, 389	:3103	SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:3104	Q.FROM.LS:
C 1C8, 38A	:3105	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1C9, 38A	:3106	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1CA, 38A	:3107	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1CB, 38A	:3108	SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3109	Q.OR.LS:
C 1CC, 38B	:3110	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1CD, 38B	:3111	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN

ZZ-ENKCH-1.0	:	ENKCH.MIC	CONTROL ROM CONTENT	H 6	19-MAY-83	Fiche 1 Frame H6	Sequence 72	
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00	Page 69
:	:	ENKCH.MIC	CONTROL ROM CONTENTS			VER 00.01		

C 1CE, 38B	:3112	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1CF, 38B	:3113	SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:3114	
	:3115	Q.AND.LS:
C 1D0, 18C	:3116	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 1D1, 18C	:3117	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 1D2, 18C	:3118	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 1D3, 18C	:3119	SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
	:3120	Q.XOR.LS:
C 1D4, 38E	:3121	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1D5, 38E	:3122	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1D6, 38E	:3123	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 1D7, 38E	:3124	SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
	:3125	MOV.Q.LS:
C 1D8, 28B	:3126	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1D9, 28B	:3127	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1DA, 28B	:3128	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 1DB, 28B	:3129	SRC/O.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
	:3130	Q.NEG.LS:
C 1DC, 28A	:3131	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1DD, 28A	:3132	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1DE, 28A	:3133	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1DF, 28A	:3134	SRC/O.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3135	
	:3136	WR.COM.LS:
C 1E0, 0CF	:3137	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 1E1, 0CF	:3138	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 1E2, 0CF	:3139	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 1E3, 0CF	:3140	SRC/O.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
	:3141	WR.NEG.LS:
C 1E4, 2CA	:3142	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1E5, 2CA	:3143	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1E6, 2CA	:3144	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1E7, 2CA	:3145	SRC/O.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3146	Q.PLUS.WR.TO.LS:
C 1E8, 008	:3147	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1E9, 008	:3148	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1EA, 008	:3149	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 1EB, 008	:3150	SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
	:3151	Q.FROM.WR.TO.LS:
C 1EC, 20A	:3152	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1ED, 20A	:3153	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1EE, 20A	:3154	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 1EF, 20A	:3155	SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
	:3156	
	:3157	DEC.LS:
C 1F0, 1CA	:3158	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1F1, 1CA	:3159	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1F2, 1CA	:3160	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 1F3, 1CA	:3161	SRC/D.O,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
	:3162	COM.LS:
C 1F4, 3CF	:3163	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 1F5, 3CF	:3164	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 1F6, 3CF	:3165	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 1F7, 3CF	:3166	SRC/D.O,ALU/R.XNOR.S,DST/NOP,CIN/CIN

C 1F8, 3C9	:3167	NEG.LS:	
C 1F9, 3C9	:3168		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1FA, 3C9	:3169		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 1FB, 3C9	:3170		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:3171		SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
	:3172	INC.LS:	
C 1FC, 3C8	:3173		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1FD, 3C8	:3174		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1FE, 3C8	:3175		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 1FF, 3C8	:3176		SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
	:3177		
	:3178	.UCODE	

```

:3179 .PAGE "DIAGNOSTIC MACROS AND DEFINITIONS"
:3180
:3181 D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
:3182
:3183
:3184 SAV.WR0=1 ; STORAGE FOR WR0
:3185 SAV.WR1=2 ; STORAGE FOR WR1
:3186 SAV.WR2=3 ; STORAGE FOR WR2
:3187 SAV.WR3=4 ; STORAGE FOR WR3
:3188 T5.SUB=5 ; RESERVED FOR COMMON SUBROUTINES
:3189 T6.SUB=6 ; RESERVED FOR COMMON SUBROUTINES
:3190 T7.SUB=7 ; RESERVED FOR COMMON SUBROUTINES
:3191 TRANSFER.POINTER=7 ; POINTER FOR TRANSFER ROUTINE
:3192 MEMORY.SIZE=7 ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:3193 ; SIZING
:3194 FPA.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:3195 UBE.FOUND=7 ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:3196 T8=8 ; TEMP LOCATION 8
:3197 T9=9 ; TEMP LOCATION 9
:3198 T10=0A ; TEMP LOCATION 10
:3199 T11=0B ; TEMP LOCATION 11
:3200 T12=0C ; TEMP LOCATION 12
:3201 T13=0D ; TEMP LOCATION 13
:3202 T14=0E ; TEMP LOCATION 14
:3203 T15=0F ; TEMP LOCATION 15
:3204 PC=10 ; MACRO PROGRAM COUNTER
:3205
:3206 WR.BACKUP=11 ; BACKUP WR FOR MOV MEM.DATA TO WR[]
:3207 MM.LASTADDR+1=12 ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:3208 ; 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:3209 #FF=13 ; MASK
:3210 #FFFF=14 ; MASK
:3211 #FF000000=15 ; MASK
:3212 #FFFFFF00=16 ; MASK
:3213 CSR0.MASK=16 ; MASK
:3214 CSR1.MASK=17 ; MASK (01003FFF)
:3215 CSR2.MASK=18 ; MASK (7FFE3FFF)
:3216 #FE7FFFFFFF=19 ; MASK
:3217 UBS.SET=1A ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:3218 UBS.DATA=1B ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:3219 ; DATA TEST
:3220
:3221 ERR.CON.NA=1E ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:3222 ; LOADING CONTROL AND STATUS REG IN INITIAL
:3223 ; TESTS
:3224 ERR.CON=1F ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:3225 ; LOADING CONTROL AND STATUS REG IN INITIAL
:3226 ; TESTS
:3227
:3228 #1=20 ; PATTERN 00000001 (HEX)
:3229 #2=21 ; PATTERN 00000002
:3230 #4=22 ; PATTERN 00000004
:3231 #8=23 ; PATTERN 00000008
:3232 #10=24 ; PATTERN 00000010
:3233 #20=25 ; PATTERN 00000020
  
```


:3234	#40=26	: PATTERN 00000040
:3235	#80=27	: PATTERN 00000080
:3236	#100=28	: PATTERN 00000100
:3237	#200=29	: PATTERN 00000200
:3238	#400=2A	: PATTERN 00000400
:3239	#800=2B	: PATTERN 00000800
:3240	#1000=2C	: PATTERN 00001000
:3241	#2000=2D	: PATTERN 00002000
:3242	#4000=2E	: PATTERN 00004000
:3243	#8000=2F	: PATTERN 00008000
:3244	#10000=30	: PATTERN 00010000
:3245	#20000=31	: PATTERN 00020000
:3246	#40000=32	: PATTERN 00040000
:3247	#80000=33	: PATTERN 00080000
:3248	#100000=34	: PATTERN 00100000
:3249	#200000=35	: PATTERN 00200000
:3250	#400000=36	: PATTERN 00400000
:3251	#800000=37	: PATTERN 00800000
:3252	#1000000=38	: PATTERN 01000000
:3253	#2000000=39	: PATTERN 02000000
:3254	#4000000=3A	: PATTERN 04000000
:3255	#8000000=3B	: PATTERN 08000000
:3256	#10000000=3C	: PATTERN 10000000
:3257	#20000000=3D	: PATTERN 20000000
:3258	#40000000=3E	: PATTERN 40000000
:3259	#80000000=3F	: PATTERN 80000000
:3260		
:3261	BIT0=20	: PATTERN 00000001
:3262	BIT1=21	: PATTERN 00000002
:3263	BIT2=22	: PATTERN 00000004
:3264	BIT3=23	: PATTERN 00000008
:3265	BIT4=24	: PATTERN 00000010
:3266	BIT5=25	: PATTERN 00000020
:3267	BIT6=26	: PATTERN 00000040
:3268	BIT7=27	: PATTERN 00000080
:3269	BIT8=28	: PATTERN 00000100
:3270	BIT9=29	: PATTERN 00000200
:3271	BIT10=2A	: PATTERN 00000400
:3272	BIT11=2B	: PATTERN 00000800
:3273	BIT12=2C	: PATTERN 00001000
:3274	BIT13=2D	: PATTERN 00002000
:3275	BIT14=2E	: PATTERN 00004000
:3276	BIT15=2F	: PATTERN 00008000
:3277	BIT16=30	: PATTERN 00010000
:3278	BIT17=31	: PATTERN 00020000
:3279	BIT18=32	: PATTERN 00040000
:3280	BIT19=33	: PATTERN 00080000
:3281	BIT20=34	: PATTERN 00100000
:3282	BIT21=35	: PATTERN 00200000
:3283	BIT22=36	: PATTERN 00400000
:3284	BIT23=37	: PATTERN 00800000
:3285	BIT24=38	: PATTERN 01000000
:3286	BIT25=39	: PATTERN 02000000
:3287	BIT26=3A	: PATTERN 04000000
:3288	BIT27=3B	: PATTERN 08000000

ZZ-ENKCH-1.0
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

MICRO2 1M(01)

DIAGNOSTIC MACROS A19-MAY-83

19-MAY-83 13:28:52

DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame L6

Sequence 76

Rev 01.00 Page 73

```
:3289          BIT28=3C          : PATTERN 10000000
:3290          BIT29=3D          : PATTERN 20000000
:3291          BIT30=3E          : PATTERN 40000000
:3292          BIT31=3F          : PATTERN 80000000
:3293
:3294
:3295          :CSR ADDRESS MNEMONICS
:3296
:3297          CSR0=4E          : ALL BITS CLEAR TO ACCESS CSR 0
:3298          CSR1=22          : BIT 2 SET TO ACCESS CSR 1
:3299          CSR2=23          : BIT 3 SET TO ACCESS CSR 2
:3300
:3301          :CONTROL AND STATUS WORD BITS
:3302
:3303          PRINT.UBE=26      : PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
:3304                                (INDICATOR IN LS 7)
:3305          PRINT.R80=27      : PRINT IF R80 PRESENT OR NOT PRESENT AND ON
:3306                                WHICH DRIVE (BIT 7). INDICATOR IN LS 7,
:3307                                DRIVE NUMBER IN LS 6.
:3308          ERROR=28          : TEST ERROR INDICATION (BIT 8)
:3309          SET.PA.ERR=29     : TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
:3310                                ADDRESS POINTED TO BY LS 7 (BIT 9)
:3311          CONSOLE.LOE=2A     : INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
:3312                                (FOR INITIAL TESTS)
:3313          PRINT.MEM.SIZE=2B : PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
:3314                                (SIZE IN LS 7)
:3315          PRINT.FPA=2C      : PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
:3316                                (INDICATOR IN LS 7)
:3317          XFER.DATA=2D       : INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
:3318          EOS=2E            : END OF SECTION (BIT 14)
:3319          BEGIN.TEST=2F     : BEGINNING OF TEST (BIT 15)
:3320          INTERRUPT.EN=30    : ENABLE INTERRUPT OR SIGNAL SPECIFIED BY BITS
:3321                                17 THRU 22 AND 28 THRU 30 (BIT 16)
:3322
:3323          CONSOLE.HALT=31     : CONSOLE HALT INTERRUPT (BIT 17)
:3324          PWR.FAIL=32        : PWR FAIL INT (BIT 18)
:3325          INTERVAL.TIM=33    : INTRVL TIM INT (BIT 19)
:3326          CONSOLE.ATTN=34    : CONSOLE ATTN INTERRUPT (BIT 20)
:3327          CONSOLE.ACK=35     : CONSOLE ACK INTERRUPT (BIT 21)
:3328          DISABLE.MEM.REF=36 : DISABLE MEMORY REFERENCES (BIT 22)
:3329          CINIT=3C          : UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
:3330          UBS.DCLO=3D        : UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
:3331          UBS.BBSY=3E        : UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
:3332
:3333          NA.EXP.REC=37      : PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
:3334          OTHER.DATA=38      : PRINT A VALUE UNDER OTHER DATA (BIT 24)
:3335          CONSOLE.LOOP=39    : CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
:3336                                COUNT IN LS (BIT 25)
:3337          PRINT.CRD=3A        : PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
:3338          CLR.PA.ERR=3B      : TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
:3339                                ADDRESS POINTED TO BY LS 7 (BIT 27)
:3340          PRINT.IDC=3F        : PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
:3341                                (INDICATOR IN LS 7)
:3342
:3343          :MODULE CODES
```

```

3344      WCS=20      : MODULE CODE FOR WCS BOARD (BIT 0 SET)
3345      CPU=21      : MODULE CODE FOR CPU BOARD (BIT 1 SET)
3346      MCT=22      : MODULE CODE FOR MCT BOARD (BIT 2 SET)
3347      FPA=23      : MODULE CODE FOR FPA BOARD (BIT 3 SET)
3348      ARRAY=24     : MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
3349      IDC=25       : MODULE CODE FOR IDC BOARD (BIT 5 SET)
3350
3351 ;CSR 1 ERROR BITS
3352
3353      VALID.ERR=2E   : VALID NOT SET IF 1 (BIT 14)
3354      TB.PAR.ERR=2F  : TB PARITY ERROR (BIT 15)
3355      NXM=30         : NON-EXISTANT MEMORY ERROR (BIT 16)
3356      UB.BSY=31      : UNIBUS BUSY (BIT 17)
3357      ADP.REG=32     : UNIBUS ADAPTER REGISTER SELECT (BIT 18)
3358      WR.ACROSS.PG=33 : WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
3359      OP.ERR=34      : OPERATION ERROR (BIT 20)
3360      TB.MISS=35     : TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
3361                      : BUFFER) (BIT 21)
3362
3363      ACCESS.REFUSED=36 : PROTECTION ERROR ON ACCESS (BIT 22)
3364      MODIFY.REFUSED=37 : PROTECTION ERROR ON WRITE (BIT 23)
3365
3366      CRD=3E         : SINGLE BIT ERROR CORRECTED (BIT 30)
3367      RDS=3F         : UNCORRECTABLE DATA ERROR (BIT 31)
3368
3369 ;CSR 1 CONTROL BITS
3370
3371      ECC.DIS=39      : CSR1 ECC DISABLE BIT (BIT 25)
3372      DIAG.CHK=3A    : CSR1 DIAG CHK BIT (BIT 26)
3373      MME=3B         : CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
3374      INH.CRD=3C     : CSR1 INHIBIT REPORT CRD BIT (BIT 28)
3375      TB.PAR.DIAG=3D : CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
3376                      : (BIT 29)
3377
3378 ;UBE CONTROL REGISTER BITS
3379
3380 ;CONTROL REG 1
3381
3382      GO=20          : START UBE (BIT 0)
3383      BR4=21         : ISSUE BUS REQUEST 4 (BIT 1)
3384      BR5=22         : ISSUE BUS REQUEST 5 (BIT 2)
3385      BR6=23         : ISSUE BUS REQUEST 6 (BIT 3)
3386      BR7=24         : ISSUE BUS REQUEST 7 (BIT 4)
3387      NPR=25         : ISSUE NPR (BIT 5)
3388      INT.ODNE=26    : INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
3389      RDY=27         : READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
3390      CO0=28         : C0 BIT (BIT 8)
3391      CO1=29         : C1 BIT (BIT 9)
3392      FUNA=2A        : FUNCTION BIT A (BIT 10)
3393      FUNB=2B        : FUNCTION BIT B (BIT 11)
3394      CC+DAT=2C      : DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
3395      INH.DATIP.ROL=2D : INHIBIT ROL ON DATIP (BIT 13)
3396      DATOB.ON.DATIP=2E : DO DATOB AFTER DATIP CYCLE (BIT 14)
3397      ERR=2F         : EXERCISOR OR UNIBUS ERROR (BIT 15)
3398
  
```



```

:3399
:3400 :CONTROL REG 2
:3401 EXT.MEM0=20 : EXTENDED MEMORY BIT 0 (BIT 0)
:3402 EXT.MEM1=21 : EXTENDED MEMORY BIT 1 (BIT 1)
:3403 INH.INC.ADDR&CC=22 : INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
:3404 INH.SACK=23 : INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)
:3405 PWR.DWN.SEQ=24 : ASSERT ACLO (BIT 4)
:3406 WNG.GNT.BCK=25 : NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
:3407 MAX.LATE.ERR=26 : NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
:3408 NO.SACK.ERR=27 : NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
:3409 NO.SSYN.ERR=28 : NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
:3410 WRG.A.LINES=29 : ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
:3411 NO.GNT+NOT.ONE.GNT=2A : NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
:3412 INTR.SSYN.ERR=2B : NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
:3413 ENB.PB=2C : ENABLE PARITY BIT B (BIT 12)
:3414 CCOVF=2D : CYCLE COUNT OVERFLOW (BIT 13)
:3415 TM.DLY=2E : REQUEST BUS EVERY 10 USEC (BIT 14)
:3416
:3417 :BG6 MNEMONIC
:3418 BG6=23 : BIT 3 SET FOR BG 6 ADDRESS
:3419
:3420 :ERROR AND CONTROL INFORMATION
:3421
:3422 CONTROL.STATUS=40 : CONTROL AND STATUS WORD
:3423 ERROR.NUMBER=41 : ERROR NUMBER OF CURRENT TEST
:3424 EXPECTED.DATA=42 : EXPECTED RESULT OF CURRENT TEST
:3425 RECEIVED.DATA=43 : ACTUAL RESULT OF CURRENT TEST
:3426 ADDRESS.DATA=44 : OTHER PERTINENT DATA
:3427 ERROR.MASK=45 : BITS TO CHECK IN RESULT
:3428 MODULE.NUM=46 : STORAGE OF MODULE NUMBER (CODED)
:3429 LOOP.CNT=47 : STORAGE OF LOOP COUNT FOR CONSOLE LOOP
:3430 CONTROL.
:3431 ERROR.CONTROL=48 : STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
:3432 LOE=20 : LOOP ON ERROR IF SET (BIT0)
:3433 NER=21 : NO ERROR REPORT IF SET (BIT1)
:3434 BELL=22 : BELL ON ERROR IF SET (BIT2)
:3435 HOE=23 : HALT ON ERROR IF SET (BIT3)
:3436 SER=24 : ENABLE ERROR REPORT ON CRD ERRORS IF SET
:3437 APT.PRESENT=25 : APT PRESENT IF SET (BIT5)
:3438 LOOP.COMMAND=26 : LOOP COMMAND TYPED IF SET (BIT 6)
:3439 SPECIAL=27 : SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
:3440
:3441 APT.PARAM=49 : APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
:3442 : WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
:3443 : AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
:3444 APT.RESERVED=4A : RESERVED FOR FUTURE APT USE
:3445
:3446 :MISCELLANEOUS CONSTANTS AND STORAGE
:3447
:3448 #AAAAAAA=4C : CONSTANT
:3449 ALTER.ODD=4C : CONSTANT
:3450 #55555555=4D : CONSTANT
:3451 ALTER.EVEN=4D : CONSTANT
:3452 #0=4E : CONSTANT
:3453 ZERO=4E : CONSTANT
    
```

ZZ-ENKCH-1.0
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

MICRO2 1M(01)

DIAGNOSTIC MACROS A19-MAY-83

19-MAY-83 13:28:52

Fiche 1 Frame B7

Sequence 79

Rev 01.00

Page 76

DIAGNOSTIC MACROS AND DEFINITIONS

ENKCH Micro-diagnostic for IDC module

```
:3454          #FFFFFFF=4F          : CONSTANT
:3455          ONES=4F             : CONSTANT
:3456          PREVIOUS.ERROR=50   : ERROR NUMBER OF LAST ERROR
:3457
:3458          DATA.1=51          : STORAGE FOR FPA DATA
:3459          DATA.2=52          : STORAGE FOR FPA DATA
:3460          DATA.3=53          : STORAGE FOR FPA DATA
:3461          FIRST.FLT=54        : STORAGE FOR FPA DATA
:3462          SEC.FLT=55          : STORAGE FOR FPA DATA
:3463          THIRD.FLT=56        : STORAGE FOR FPA DATA
:3464          T57=57              : TEMP LOCATION 57
:3465          T58=58              : TEMP LOCATION 58
:3466          T59=59              : TEMP LOCATION 59
:3467
:3468          BEDB=5A              :UBE (BUS EXERCISOR) DATA BUF REG
:3469          BECC=5B              :UBE (BUS EXERCISOR) CYCLE COUNT REG
:3470          BEBA=5C              :UBE (BUS EXERCISOR) ADDR REG
:3471          BECR1=5D             :UBE (BUS EXERCISOR) CONTROL REG 1
:3472          CLEAR.ERR.ADDR=5E    :UBE CLEAR ERROR REG ADDRESS
:3473          BECR2=5F             :UBE (BUS EXERCISOR) CONTROL REG 2
:3474
:3475          #3(H)=60             : CONSTANT
:3476          #12(H)=61           : CONSTANT
:3477          #33333333=62        : CONSTANT
:3478          #0F0F0F0F=63        : CONSTANT
:3479          #00FF00FF=64        : CONSTANT
:3480          #162(H)=65          : CONSTANT FOR LS TRANSFER POINTER
:3481          BR7.IDENT=66         : BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:3482          BG7=67              : BG7 ADDRESS (BITS 2 AND 3 SET)
:3483
:3484          :TEMPS FOR CPU TESTS
:3485          L30=30              :LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:3486          L31=31              :LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:3487          L53=53              :LS 53 TEMP
:3488          L73=73              :LS 73 TEMP
:3489
:3490          :REGISTER ADDRESSES
:3491
:3492          CONTROL.OS=78         : HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:3493          SHIFT.OS(3-0)=79     : 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3494          : (LS 20-2F)
:3495          SHIFT.OS(4-0)=7B     : 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3496          : (LS 20-3F)
:3497          OS=7C                 : OS REGISTER
:3498          DEC.CON=7D            : DECIMAL CARRIES
:3499          SIZE=7E              : SIZE REGISTER
:3500          MDT= 7E              : MEMORY REFERENCE DATA SIZE
:3501          INTERRUPT.VEC=7E     : HARDWARE INTERRUPT VECTOR
:3502          :
:3503          : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:3504          : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:3505          :
:3506          PSL.CC=7F            : PSL CONDITION CODES
```

```

:3507 .PAGE
:3508 XD.ADRS/= <16:9> , .VALIDITY=<XD.VAL>
:3509
:3510
:3511 SAV.WR0=1 : STORAGE FOR WR0
:3512 SAV.WR1=2 : STORAGE FOR WR1
:3513 SAV.WR2=3 : STORAGE FOR WR2
:3514 SAV.WR3=4 : STORAGE FOR WR3
:3515 T5.SUB=5 : RESERVED FOR COMMON SUBROUTINES
:3516 T6.SUB=6 : RESERVED FOR COMMON SUBROUTINES
:3517 T7.SUB=7 : RESERVED FOR COMMON SUBROUTINES
:3518 TRANSFER.POINTER=7 : POINTER FOR TRANSFER ROUTINE
:3519 MEMORY.SIZE=7 : STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
:3520 : SIZING
:3521 FPA.FOUND=7 : STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
:3522 UBE.FOUND=7 : STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
:3523 T8=8 : TEMP LOCATION 8
:3524 T9=9 : TEMP LOCATION 9
:3525 T10=0A : TEMP LOCATION 10
:3526 T11=0B : TEMP LOCATION 11
:3527 T12=0C : TEMP LOCATION 12
:3528 T13=0D : TEMP LOCATION 13
:3529 T14=0E : TEMP LOCATION 14
:3530 T15=0F : TEMP LOCATION 15
:3531 PC=10 : MACRO PROGRAM COUNTER
:3532
:3533 WR.BACKUP=11 : BACKUP WR FOR MOV MEM.DATA TO WR[]
:3534 MM.LASTADDR+1=12 : STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
:3535 : 1 (FIRST NON-EXISTANT MEMORY ADDRESS)
:3536 #FF=13 : MASK
:3537 #FFFF=14 : MASK
:3538 #FFF00000=15 : MASK
:3539 #FFFFFF00=16 : MASK
:3540 CSR0.MASK=16 : MASK
:3541 CSR1.MASK=17 : MASK (01003FFF)
:3542 CSR2.MASK=18 : MASK (FFFE3FFF)
:3543 #FE7FFFFFFF=19 : MASK
:3544 UBS.SET=1A : CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
:3545 UBS.DATA=1B : MASK (7FFF8000) USED AS ERROR MASK FOR UBS
:3546 : DATA TEST
:3547
:3548 ERR.CON.NA=1E : CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
:3549 : LOADING CONTROL AND STATUS REG IN INITIAL
:3550 : TESTS
:3551 ERR.CON=1F : CONSTANT OF 500(H) BITS 10 & 8 SET FOR
:3552 : LOADING CONTROL AND STATUS REG IN INITIAL
:3553 : TESTS
:3554
:3555 #1=20 : PATTERN 00000001 (HEX)
:3556 #2=21 : PATTERN 00000002
:3557 #4=22 : PATTERN 00000004
:3558 #8=23 : PATTERN 00000008
:3559 #10=24 : PATTERN 00000010
:3560 #20=25 : PATTERN 00000020
:3561 #40=26 : PATTERN 00000040
    
```


:3562	#80=27	: PATTERN 00000080
:3563	#100=28	: PATTERN 00000100
:3564	#200=29	: PATTERN 00000200
:3565	#400=2A	: PATTERN 00000400
:3566	#800=2B	: PATTERN 00000800
:3567	#1000=2C	: PATTERN 00001000
:3568	#2000=2D	: PATTERN 00002000
:3569	#4000=2E	: PATTERN 00004000
:3570	#8000=2F	: PATTERN 00008000
:3571	#10000=30	: PATTERN 00010000
:3572	#20000=31	: PATTERN 00020000
:3573	#40000=32	: PATTERN 00040000
:3574	#80000=33	: PATTERN 00080000
:3575	#100000=34	: PATTERN 00100000
:3576	#200000=35	: PATTERN 00200000
:3577	#400000=36	: PATTERN 00400000
:3578	#800000=37	: PATTERN 00800000
:3579	#1000000=38	: PATTERN 01000000
:3580	#2000000=39	: PATTERN 02000000
:3581	#4000000=3A	: PATTERN 04000000
:3582	#8000000=3B	: PATTERN 08000000
:3583	#10000000=3C	: PATTERN 10000000
:3584	#20000000=3D	: PATTERN 20000000
:3585	#40000000=3E	: PATTERN 40000000
:3586	#80000000=3F	: PATTERN 80000000
:3587		
:3588	BIT0=20	: PATTERN 00000001
:3589	BIT1=21	: PATTERN 00000002
:3590	BIT2=22	: PATTERN 00000004
:3591	BIT3=23	: PATTERN 00000008
:3592	BIT4=24	: PATTERN 00000010
:3593	BIT5=25	: PATTERN 00000020
:3594	BIT6=26	: PATTERN 00000040
:3595	BIT7=27	: PATTERN 00000080
:3596	BIT8=28	: PATTERN 00000100
:3597	BIT9=29	: PATTERN 00000200
:3598	BIT10=2A	: PATTERN 00000400
:3599	BIT11=2B	: PATTERN 00000800
:3600	BIT12=2C	: PATTERN 00001000
:3601	BIT13=2D	: PATTERN 00002000
:3602	BIT14=2E	: PATTERN 00004000
:3603	BIT15=2F	: PATTERN 00008000
:3604	BIT16=30	: PATTERN 00010000
:3605	BIT17=31	: PATTERN 00020000
:3606	BIT18=32	: PATTERN 00040000
:3607	BIT19=33	: PATTERN 00080000
:3608	BIT20=34	: PATTERN 00100000
:3609	BIT21=35	: PATTERN 00200000
:3610	BIT22=36	: PATTERN 00400000
:3611	BIT23=37	: PATTERN 00800000
:3612	BIT24=38	: PATTERN 01000000
:3613	BIT25=39	: PATTERN 02000000
:3614	BIT26=3A	: PATTERN 04000000
:3615	BIT27=3B	: PATTERN 08000000
:3616	BIT28=3C	: PATTERN 10000000

```

3617 BIT29=3D ; PATTERN 20000000
3618 BIT30=3E ; PATTERN 40000000
3619 BIT31=3F ; PATTERN 80000000
3620
3621
3622 :CSR ADDRESS MNEMONICS
3623
3624 CSR0=4E ; ALL BITS CLEAR TO ACCESS CSR 0
3625 CSR1=22 ; BIT 2 SET TO ACCESS CSR 1
3626 CSR2=23 ; BIT 3 SET TO ACCESS CSR 2
3627
3628
3629 :CONTROL AND STATUS WORD BITS
3630
3631 PRINT.UBE=26 ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
3632 ; (INDICATOR IN LS 7)
3633 PRINT.R80=27 ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
3634 ; WHICH DRIVE (BIT 7). INDICATOR IN LS 7,
3635 ; DRIVE NUMBER IN LS 6.
3636 ERROR=28 ; TEST ERROR INDICATION (BIT 8)
3637 SET.PA.ERR=29 ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
3638 ; ADDRESS POINTED TO BY LS 7 (BIT 9)
3639 CONSOLE.LOE=2A ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
3640 ; (FOR INITIAL TESTS)
3641 PRINT.MEM.SIZE=2B ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
3642 ; (SIZE IN LS 7)
3643 PRINT.FPA=2C ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
3644 ; (INDICATOR IN LS 7)
3645 XFER.DATA=2D ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
3646 EOS=2E ; END OF SECTION (BIT 14)
3647 BEGIN.TEST=2F ; BEGINNING OF TEST (BIT 15)
3648 INTERRUPT.EN=30 ; ENABLE INTERRUPT OR SIGNAL SPECIFIED BY BITS
3649 ; 17 THRU 22 AND 28 THRU 30 (BIT 16)
3650
3651 CONSOLE.HALT=31 ; CONSOLE HALT INTERRUPT (BIT 17)
3652 PWR.FAIL=32 ; PWR FAIL INT (BIT 18)
3653 INTERVAL.TIM=33 ; INTRVL TIM INT (BIT 19)
3654 CONSOLE.ATTN=34 ; CONSOLE ATTN INTERRUPT (BIT 20)
3655 CONSOLE.ACK=35 ; CONSOLE ACK INTERRUPT (BIT 21)
3656 DISABLE.MEM.REF=36 ; DISABLE MEMORY REFERENCES (BIT 22)
3657 CINIT=3C ; UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
3658 UBS.DCLO=3D ; UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
3659 UBS.BBSY=3E ; UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
3660
3661 NA.EXP.REC=37 ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
3662 OTHER.DATA=38 ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
3663 CONSOLE.LOOP=39 ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
3664 ; COUNT IN LS (BIT 25)
3665 PRINT.CRD=3A ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
3666 CLR.PA.ERR=3B ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
3667 ; ADDRESS POINTED TO BY LS 7 (BIT 27)
3668 PRINT.IDC=3F ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
3669 ; (INDICATOR IN LS 7)
3670
3671

```

```

:3672 ;MODULE CODES
:3673
:3674     WCS=20      ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
:3675     CPU=21     ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
:3676     MCT=22     ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
:3677     FPA=23     ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
:3678     ARRAY=24   ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
:3679     IDC=25     ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
:3680
:3681 ;CSR 1 ERROR BITS
:3682
:3683     VALID.ERR=2E ; VALID NOT SET IF 1 (BIT 14)
:3684     TB.PAR.ERR=2F ; TB PARITY ERROR (BIT 15)
:3685     NXM=30       ; NON-EXISTANT MEMORY ERROR (BIT 16)
:3686     UB.BSY=31    ; UNIBUS BUSY (BIT 17)
:3687     ADP.REG=32    ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
:3688     WR.ACROSS.PG=33 ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
:3689     OP.ERR=34     ; OPERATION ERROR (BIT 20)
:3690     TB.MISS=35    ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
:3691                     ; BUFFER) (BIT 21)
:3692     ACCESS.REFUSED=36 ; PROTECTION ERROR ON ACCESS (BIT 22)
:3693     MODIFY.REFUSED=37 ; PROTECTION ERROR ON WRITE (BIT 23)
:3694
:3695     CRD=3E         ; SINGLE BIT ERROR CORRECTED (BIT 30)
:3696     RDS=3F         ; UNCORRECTABLE DATA ERROR (BIT 31)
:3697
:3698 ;CSR 1 CONTROL BITS
:3699
:3700
:3701     ECC.DIS=39     ; CSR1 ECC DISABLE BIT (BIT 25)
:3702     DIAG.CHK=3A    ; CSR1 DIAG CHK BIT (BIT 26)
:3703     MME=3B         ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
:3704     INH.CRD=3C     ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
:3705     TB.PAR.DIAG=3D ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
:3706                     ; (BIT 29)
:3707
:3708 ;UBE CONTROL REGISTER BITS
:3709
:3710 ;CONTROL REG 1
:3711
:3712     GO=20          ; START UBE (BIT 0)
:3713     BR4=21         ; ISSUE BUS REQUEST 4 (BIT 1)
:3714     BR5=22         ; ISSUE BUS REQUEST 5 (BIT 2)
:3715     BR6=23         ; ISSUE BUS REQUEST 6 (BIT 3)
:3716     BR7=24         ; ISSUE BUS REQUEST 7 (BIT 4)
:3717     NPR=25         ; ISSUE NPR (BIT 5)
:3718     INT.ODNE=26    ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
:3719     RDY=27         ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
:3720     CO0=28         ; C0 BIT (BIT 8)
:3721     CO1=29         ; C1 BIT (BIT 9)
:3722     FUNA=2A        ; FUNCTION BIT A (BIT 10)
:3723     FUNB=2B        ; FUNCTION BIT B (BIT 11)
:3724     CC+DAT=2C      ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
:3725     INH.DATIP.ROL=2D ; INHIBIT ROL ON DATIP (BIT 13)
:3726     DATOB.ON.DATIP=2E ; DO DATOB AFTER DATIP CYCLE (BIT 14)
    
```


ZZ-ENKCH-1.0 :
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

DIAGNOSTIC MACROS A19-MAY-83
MICRO2 1M(01) 19-MAY-83 13:28:52
DIAGNOSTIC MACROS AND DEFINITIONS

Fiche 1 Frame G7
ENKCH Micro-diagnostic for IDC module

Sequence 84
Rev 01.00

Page 81

```
:3727      ERR=2F      ; EXERCISOR OR UNIBUS ERROR (BIT 15)
:3728
:3729      ;CONTROL REG 2
:3730      EXT.MEM0=20      ; EXTENDED MEMORY BIT 0 (BIT 0)
:3731      EXT.MEM1=21      ; EXTENDED MEMORY BIT 1 (BIT 1)
:3732      INH.INC.ADDR&CC=22 ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
:3733      INH.SACK=23      ; INHIBIT ASSERTING BUS SACK ON BUS GRANT (BIT 3)
:3734      PWR.DWN.SEQ=24    ; ASSERT ACLO (BIT 4)
:3735      WNG.GNT.BCK=25    ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
:3736      MAX.LATE.ERR=26   ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
:3737      NO.SACK.ERR=27    ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
:3738      NO.SSYN.ERR=28    ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
:3739      WRG.A.LINES=29    ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
:3740      NO.GNT+NOT.ONE.GNT=2A ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
:3741      INTR.SSYN.ERR=2B  ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
:3742      ENB.PB=2C        ; ENABLE PARITY BIT B (BIT 12)
:3743      CCOVF=2D         ; CYCLE COUNT OVERFLOW (BIT 13)
:3744      TM.DLY=2E        ; REQUEST BUS EVERY 10 USEC (BIT 14)
:3745
:3746      ;BG6 MNEMONIC
:3747      BG6=23           ; BIT 3 SET FOR BG 6 ADDRESS
:3748
:3749      ;ERROR AND CONTROL INFORMATION
:3750
:3751      CONTROL.STATUS=40  ; CONTROL AND STATUS WORD
:3752      ERROR.NUMBER=41   ; ERROR NUMBER OF CURRENT TEST
:3753      EXPECTED.DATA=42  ; EXPECTED RESULT OF CURRENT TEST
:3754      RECEIVED.DATA=43  ; ACTUAL RESULT OF CURRENT TEST
:3755      ADDRESS.DATA=44   ; OTHER PERTINENT DATA
:3756      ERROR.MASK=45     ; BITS TO CHECK IN RESULT
:3757      MODULE.NUM=46     ; STORAGE OF MODULE NUMBER (CODED)
:3758      LOOP.CNT=47      ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
:3759      ;CONTROL
:3760      ERROR.CONTROL=48  ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
:3761      LOE=20           ; LOOP ON ERROR IF SET (BIT0)
:3762      NER=21           ; NO ERROR REPORT IF SET (BIT1)
:3763      BELL=22          ; BELL ON ERROR IF SET (BIT2)
:3764      HOE=23           ; HALT ON ERROR IF SET (BIT3)
:3765      SER=24           ; ENABLE ERROR REPORT ON CRD ERRORS IF SET
:3766      APT.PRESENT=25    ; APT PRESENT IF SET (BIT5)
:3767      LOOP.COMMAND=26   ; LOOP COMMAND TYPED IF SET (BIT 6)
:3768      SPECIAL=27       ; SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
:3769
:3770      APT.PARAM=49       ; APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
:3771                          ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
:3772                          ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
:3773      APT.RESERVED=4A     ; RESERVED FOR FUTURE APT USE
:3774
:3775      ;MISCELLANEOUS CONSTANTS AND STORAGE
:3776
:3777      #AAAAAAA=4C        ; CONSTANT
:3778      ALTER.ODD=4C       ; CONSTANT
:3779      #55555555=4D       ; CONSTANT
:3780      ALTER.EVEN=4D      ; CONSTANT
:3781      #0=4E              ; CONSTANT
```

```

:3782      ZERO=4E      : CONSTANT
:3783      #FFFFFFF=4F  : CONSTANT
:3784      ONES=4F      : CONSTANT
:3785      PREVIOUS.ERROR=50 : ERROR NUMBER OF LAST ERROR
:3786
:3787      DATA.1=51    : STORAGE FOR FPA DATA
:3788      DATA.2=52    : STORAGE FOR FPA DATA
:3789      DATA.3=53    : STORAGE FOR FPA DATA
:3790      FIRST.FLT=54  : STORAGE FOR FPA DATA
:3791      SEC.FLT=55    : STORAGE FOR FPA DATA
:3792      THIRD.FLT=56  : STORAGE FOR FPA DATA
:3793      T57=57        : TEMP LOCATION 57
:3794      T58=58        : TEMP LOCATION 58
:3795      T59=59        : TEMP LOCATION 59
:3796
:3797      BEDB=5A        :UBE (BUS EXERCISOR) DATA BUF REG
:3798      BECC=5B        :UBE (BUS EXERCISOR) CYCLE COUNT REG
:3799      BEBA=5C        :UBE (BUS EXERCISOR) ADDR REG
:3800      BECR1=5D       :UBE (BUS EXERCISOR) CONTROL REG 1
:3801      CLEAR.ERR.ADDR=5E :UBE CLEAR ERROR REG ADDRESS
:3802      BECR2=5F       :UBE (BUS EXERCISOR) CONTROL REG 2
:3803
:3804      #3(H)=60       : CONSTANT
:3805      #12(H)=61      : CONSTANT
:3806      #33333333=62   : CONSTANT
:3807      #0F0F0F0F=63   : CONSTANT
:3808      #00FF00FF=64   : CONSTANT
:3809      #162(H)=65     : CONSTANT FOR LS TRANSFER POINTER
:3810      BR7.IDENT=66   : BR7 IDENTIFIER (1010(B) IN BITS 5-2)
:3811      BG7=67        : BG7 ADDRESS (BITS 2 AND 3 SET)
:3812
:3813      :TEMPS FOR CPU TESTS
:3814      L30=30         :LS 30 TEMP (RESTORED TO 10000 AFTER USE)
:3815      L31=31         :LS 31 TEMP (RESTORE TO 20000 AFTER USE)
:3816      L53=53        :LS 53 TEMP
:3817      L73=73        :LS 73 TEMP
:3818
:3819      :REGISTER ADDRESSES
:3820
:3821      CONTROL.OS=78    : HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
:3822      SHIFT.OS(3-0)=79 : 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3823                          (LS 20-2F)
:3824      SHIFT.OS(4-0)=7B : 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
:3825                          (LS 20-3F)
:3826      OS=7C           : OS REGISTER
:3827      DEC.CON=7D      : DECIMAL CARRIES
:3828      SIZE=7E         : SIZE REGISTER
:3829      MDT= 7E        : MEMORY REFERENCE DATA SIZE
:3830      INTERRUPT.VEC=7E : HARDWARE INTERRUPT VECTOR
:3831
:3832      :
:3833      : THIS WORD IS THE HARDWARE CONDITION CODES. THE CC'S CAN BE READ
:3834      : OR ARE WRITTEN HERE. NO OTHER BITS IN THE PSL ARE EFFECTED.
:3835      PSL.CC=7F        : PSL CONDITION CODES
:3836
    
```

```

3837
3838 .TOC "COMMON LS ASSIGNMENTS (TO REGISTERS)"
3839
3840 : UPPER HALF OF LS
3841 :
3842 : XGPR.OS=0F8 : HARDWARE INDEX TO XGPRS
3843 : USER.INDEX(3-0)=0F9 : 16 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
3844 : : AREA (LS LOCATIONS 0A0 THROUGH 0AF)
3845 : USER.INDEX(4-0)=0FB : 32 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
3846 : : AREA (LS LOCATIONS 0A0 THROUGH 0BF)
3847 : CCR=0FC : CONSOLE READ REGISTER
3848 : TIMER=0FD : INTERVAL TIMER VALUE.
3849 : ALU.CC=0FE : ALU CONDITION CODES
3850 :
3851 : THIS LOCATION IS A WRITE ONLY REGISTER. THE FOLLOWING BITS IN THE
3852 : PSL ARE AFFECTED:
3853 :
3854 : COMPATIBILITY MODE - BIT<31>
3855 : CURRENT MODE - BITS<25:24>
3856 : INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
3857 : TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
3858 : NEGATIVE CONDITION CODE - BIT<3>
3859 : ZERO CONDITION CODE - BIT<2>
3860 : OVERFLOW CONDITION CODE - BIT<1>
3861 : CARRY CONDITION CODE - BIT<0>
3862 :
3863 : PSL.HW=OFF : HARDWARE PSL BITS
3864 :
3865 :
3866 : These addresses are only accessible with the MOV micro instruction
3867 : starting at address 80(H).
3868 :
3869 : SETCSR.F0=80 : SET IDC CSR FUNCTION BITS =0 & SET WRT INHIBIT
3870 : SETCSR.F1=81 : SET IDC CSR FUNCTION BITS =1 & SET WRT INHIBIT
3871 : SETCSR.F2=82 : SET IDC CSR FUNCTION BITS =2 & SET WRT INHIBIT
3872 : SETCSR.F3=83 : SET IDC CSR FUNCTION BITS =3 & SET WRT INHIBIT
3873 : SETCSR.F4=84 : SET IDC CSR FUNCTION BITS =4 & SET WRT INHIBIT
3874 : SETCSR.F5=85 : SET IDC CSR FUNCTION BITS =5 & SET WRT INHIBIT
3875 : SETCSR.F6=86 : SET IDC CSR FUNCTION BITS =6 & SET WRT INHIBIT
3876 : SETCSR.F7=87 : SET IDC CSR FUNCTION BITS =7 & SET WRT INHIBIT
3877 : SETCSR.IE=88 : SET INTERRUPT ENABLE IN THE CSR
3878 : SETCSR.CRDY=89 : SET BIT <7> CRDY IN THE CSR
3879 : SETCSR.DS0=8A : SET CSR DRIVE SELECT BITS = 0
3880 : SETCSR.DS1=8B : SET CSR DRIVE SELECT BITS = 1
3881 : SETCSR.DS2=8C : SET CSR DRIVE SELECT BITS = 2
3882 : SETCSR.DS3=8D : SET CSR DRIVE SELECT BITS = 3
3883 : SETCSR.SSEI=8E : SET SKIP SECTOR ERROR INHIBIT IN THE CSR
3884 : SETCSR.SSFLG=8F : SET SKIP SECTOR FLAG IN THE CSR
3885 : SETCSR.MAINT=90 : SET THE MAINTENANCE BIT IN THE CSR
3886 :
3887 : CSR.MASK=91 : MASK TO CHECK WRITABLE BITS OF THE CSR
3888 : C53FFC31.MASK=91 :
3889 :
3890 : SAV.DATA.PAT=92 : LOCATION TO STORE THE CURRENT DATA PATTERN
3891 :
    
```


:3892	CSR.DAT.PAT1=93	:DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3893	CSR.DAT.PAT2=94	:DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3894	CSR.DAT.PAT3=95	:DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3895	CSR.DAT.PAT4=96	:DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3896	CSR.DAT.PAT5=97	:DATA PATTERN TO TEST THE IDC CSR WRITABLE BITS
:3897		
:3898	#1A=98	:CONSTANT
:3899		
:3900		
:3901	PORT0=99	:RESERVED FOR PORT DATA
:3902	PORT1=9A	:RESERVED FOR PORT DATA
:3903	PORT2=9B	:RESERVED FOR PORT DATA
:3904	PORT3=9C	:RESERVED FOR PORT DATA
:3905	PORT4=9D	:RESERVED FOR PORT DATA
:3906	PORT5=9E	:RESERVED FOR PORT DATA
:3907	PORT6=9F	:RESERVED FOR PORT DATA
:3908	PORT7=0A0	:RESERVED FOR PORT DATA
:3909	PORT8=0A1	:RESERVED FOR PORT DATA
:3910	PORT9=0A2	:RESERVED FOR PORT DATA
:3911	PORT10=0A3	:RESERVED FOR PORT DATA
:3912	PORT11=0A4	:RESERVED FOR PORT DATA
:3913	PORT12=0A5	:RESERVED FOR PORT DATA
:3914	PORT13=0A6	:RESERVED FOR PORT DATA
:3915	PORT14=0A7	:RESERVED FOR PORT DATA
:3916	PORT15=0A8	:RESERVED FOR PORT DATA
:3917	PORT16=0A9	:RESERVED FOR PORT DATA
:3918	PORT17=0AA	:RESERVED FOR PORT DATA
:3919	PORT18=0AB	:RESERVED FOR PORT DATA
:3920	PORT19=0AC	:RESERVED FOR PORT DATA
:3921	PORT20=0AD	:RESERVED FOR PORT DATA
:3922	PORT21=0AE	:RESERVED FOR PORT DATA
:3923	PORT22=0AF	:RESERVED FOR PORT DATA
:3924	PORT23=0B0	:RESERVED FOR PORT DATA
:3925	PORT24=0B1	:RESERVED FOR PORT DATA
:3926	PORT25=0B2	:RESERVED FOR PORT DATA
:3927	PORT26=0B3	:RESERVED FOR PORT DATA
:3928	PORT27=0B4	:RESERVED FOR PORT DATA
:3929	PORT28=0B5	:RESERVED FOR PORT DATA
:3930	PORT29=0B6	:RESERVED FOR PORT DATA
:3931	PORT30=0B7	:RESERVED FOR PORT DATA
:3932	PORT31=0B8	:RESERVED FOR PORT DATA
:3933	#0000FFFF=0B9	:CONSTANT
:3934	FORCE.IDLE=0BA	:DATA TO FORCE IDC FROM DIAG.IDLE TO IDLE
:3935	#FFF80000=0BB	:MASK FOR DAR..LOOKS AT BITS <18:0>
:3936	DAR.MASK=0BB	
:3937	CSR.PAT.1=0BC	:DATA PAT 1 TO TEST CSR
:3938	CSR.PAT.2=0BD	:DATA PAT 2 TO TEST CSR
:3939	CSR.PAT.3=0BE	:DATA PAT 3 TO TEST CSR
:3940	CSR.PAT.4=0BF	:DATA PAT 4 TO TEST CSR
:3941	CSR.PAT.5=0C0	:DATA PAT 5 TO TEST CSR
:3942	BIT7.MASK=0C1	:MASK TO CHECK ONLY BIT 7
:3943	BIT22.MASK=0C2	:MASK TO CHECK ONLY BIT 22
:3944	BIT23.MASK=0C3	:MASK TO CHECK ONLY BIT 23
:3945	#3=0C4	:CONSTANT OF 3
:3946	#FE=0C5	:CONSTANT OF FE

:3947	#1FE=0C6	:CONSTANT OF 1FE
:3948	DRIVE.STATUS=0C7	:LOCATION CONTAINS DRIVE INFORMATION
:3949	#FFFFFFF0=0C8	:MASK FOR BITS<3:0>
:3950	#6=0C9	:CONSTANT OF 6
:3951	#7FFFF=0CA	:CONSTANT
:3952	#F=0CB	:CONSTANT
:3953	#FFFFFFCF=0CC	:CONSTANT
:3954	#FFFFFFFC=0CD	:CONSTANT
:3955	#FFFFFF80=0CE	:CONSTANT
:3956	SETCSR.R80=0CF	:SETS BIT 29
:3957	#20100500=0D0	:CONSTANT
:3958	#A0100500=0D1	:CONSTANT
:3959	#FFCFFFFFF=0D2	:MASK TO CHECK BITS<21:20>
:3960	#FFFFE000=0D3	:MASK TO CHECK BITS<12:0>
:3961	#69=0D4	:CONSTANT = 105
:3962	#6837=0D5	:CONSTANT
:3963	#7200=0D6	:CONSTANT
:3964	#FFFFFF800=0D7	:MASK TO CHECK BITS <10:0>
:3965	#1021=0D8	:CONSTANT
:3966	#A0010500=0D9	:CONSTANT
:3967	#20010500=0DA	:CONSTANT
:3968	#1F=0DB	:CONSTANT
:3969	#169=0DC	:CONSTANT
:3970	#101=0DD	:CONSTANT
:3971	#FFF0FFFF=0DE	:MASK TO CHECK BITS <19:16>
:3972	#30000=0DF	:CONSTANT
:3973	#70000=0E0	:CONSTANT
:3974	#F0000=0E1	:CONSTANT
:3975	#FFFFEBFF=0E2	:MASK TO CHECK BITS <12> <10>
:3976	#FFFF6BFF=0E3	:MASK TO CHECK BITS <15> <12> <10>
:3977	#9400=0E4	:CONSTANT
:3978	#FFFF6FFF=0E5	:MASK TO CHECK BITS <15> <12>
:3979	#9000=0E6	:CONSTANT
:3980	#FFFFEFFF=0E7	:MASK TO CHECK BIT <12>
:3981	#11111111=0E8	:CONSTANT
:3982	#22222222=0E9	:CONSTANT
:3983	#44444444=0EA	:CONSTANT
:3984	#000F0000=0EB	:CONSTANT
:3985	#2D35E=0EC	:CONSTANT
:3986	#28AEE=0ED	:CONSTANT
:3987	#5A6BF=0EE	:CONSTANT
:3988	#5A6C2=0EF	:CONSTANT
:3989	#FFFFFBFF=0F0	:MASK TO CHECK BIT <10>
:3990	#FFFFFFF7F=0F1	:MASK TO CHECK BIT <7>
:3991	#10000080=0F2	:BITS TO DISABLE TIMEOUTS AND SET CSR<CRDY>
:3992	#B4D84=0F3	:WAIT LOOP TO WAIT 400ms
:3993	#FF3FFFFFF=0F4	:MASK TO CHECK <23:22>
:3994	#FFFFFFE3=0F5	:MASK TO CHECK <4:2>
:3995	#FEFFFFFF=0F6	:MASK TO CHECK <24>
:3996		
:3997		

3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047

page 'Microinstruction Formats'

The following is taken directly from the NEBULA Hardware Specification.

MICROINSTRUCTION FORMATS

1. BASIC

22	21	16	15	9	8	7	6	5	4	0
+-----+-----+-----+-----+-----+-----+										
11		DP		D		ADRS		B		CC
+-----+-----+-----+-----+-----+-----+										
SCTL										

2. MOVE

22	20	19	17	16	9	8	7	6	5	4	0
+-----+-----+-----+-----+-----+-----+											
0		1		MDP		XD		ADRS		B	
+-----+-----+-----+-----+-----+-----+											
CC SCTL											

3. EXTENDED

22	20	19	14	13	11	9	8	7	6	5	4	0
+-----+-----+-----+-----+-----+-----+												
0		1		0		XDP		SI		A		B
+-----+-----+-----+-----+-----+-----+												
CC SCTL												

4. MEM.REQ

22	19	18	16	15	9	8	7	6	5	4	0
+-----+-----+-----+-----+-----+-----+											
0		0		1		MF1		D		ADRS	
+-----+-----+-----+-----+-----+-----+											
MF2		DT		SCTL							

5. MISC

22	19	18	16	15	12	11	9	8	7	6	5	4	0
+-----+-----+-----+-----+-----+-----+													
0		0		1		0		P		0		0	
+-----+-----+-----+-----+-----+-----+													
M1		M2		0		R		0		0		SCTL	

6. JUMP

22	19	18	17	4	3	0
+-----+-----+-----+-----+-----+-----+						
0		0		0		1
+-----+-----+-----+-----+-----+-----+						
OS				JUMP ADDRESS		JCTL

7. DECODE

22	19	18	17	12	11	9	8	7	6	5	4	3	0
+-----+-----+-----+-----+-----+-----+													
0		0		0		0		DEC.ADRS		IFUNC		JCTL	
+-----+-----+-----+-----+-----+-----+													

BACK UP PC				IB REQ---				---OPC/SPEC			
SEL CM HI				IR BYTE---				---LOAD RDEST			
				LOAD OS---							


```

:4048 :.page
:4049 :
:4050 :
:4051 :1. "BASIC" Microinstruction
:4052 :
:4053 :CSR<22> = 1 (OPCODE)
:4054 :
:4055 :    This opcode bit causes LS Adrs <7> to be cleared.
:4056 :
:4057 :CSR<21:16> (Data Path Control)
:4058 :
:4059 :    This field controls the data path. It controls the 2901 ALU,
:4060 :    the ALU data source, the data destination in the 2901 array,
:4061 :    and the write to the Local Store.
:4062 :
:4063 :CSR<15:9> (D Adrs <6:0>)
:4064 :
:4065 :    This field selects which of the 128 accessible Local Store
:4066 :    locations to use.
:4067 :
:4068 :CSR<8:7> (B Adrs)
:4069 :
:4070 :    This field selects one of the four Working Registers.
:4071 :
:4072 :CSR<6:5> (Condition Codes)
:4073 :
:4074 :    This field specifies the data type and condition code control.
:4075 :
:4076 :CSR<4:0> (SCTL)
:4077 :
:4078 :    This field specifies the skip condition/micro sequencer
:4079 :    function.
    
```

```

:4080 .page
:4081
:4082
:4083 2. "MOVE" Microinstruction
:4084
:4085     CSR<22:20> = 011 (OPCODE)
:4086
:4087     This combination of opcode bits allows CSR<16> to control
:4088     LS Adrs <7>. This permits access of LS locations 80 - FF.
:4089
:4090     CSR<19:17> (Data Path Control)
:4091
:4092     This field is decoded to produce some data path control
:4093     information.
:4094
:4095         0 LS[XD] <- WR[B], WR[B] <- MEM DATA* 4 LS[XD] <- MEM DATA*
:4096         1 MEMORY <- LS [XD]*                    5 LS[XD] <- ACC/PORT
:4097         2 Q <- XWR[B]                             6 LS[XD] <- WR[B] + OS
:4098         3 WR[B] <- LS [XD]                         7 LS[XD] <- WR[B]
:4099
:4100     CSR<16:9> (XD<7:0>)
:4101
:4102     This field specifies which of the 256 Local Store locations to
:4103     use.
:4104
:4105     CSR<8:7> (B Adrs)
:4106
:4107     This field specifies which of the four Working Registers to
:4108     use. For MDP = 2, this field selects either the Backup PC or
:4109     the VAR.
:4110
:4111     CSR<6:5> (CC)
:4112
:4113     This field specifies the data type and condition code control.
:4114
:4115     CSR<4:0> (SCTL)
:4116
:4117     This field specifies the Skip control. See Table 3.
:4118
:4119
:4120 * For information on which Working Registers and Local Store locations
:4121 may be used for Memory Addresses and data source/destinations, see
:4122 the Memory Management documentation. Note that these restrictions
:4123 are due to microcoding conventions and not hardware.
    
```

:4124
:4125
:4126
:4127
:4128
:4129
:4130
:4131
:4132
:4133
:4134
:4135
:4136
:4137
:4138
:4139
:4140
:4141
:4142
:4143
:4144
:4145
:4146
:4147
:4148
:4149
:4150
:4151
:4152
:4153
:4154
:4155
:4156
:4157
:4158
:4159
:4160
:4161
:4162
:4163
:4164

page

3. "EXTENDED" Microinstruction

CSR<22:20> = 010 (OPCODE)

This opcode causes a function internal to the 2901 array; that is, an operation involving only Working Registers and the Q-Register.

CSR<19:14> (Data Path Control)

This field is decoded to supply the 2901 ALU function code, the ALU Source control, the destination code, and the ALU Carry-In.

CSR<13:11> (Shift Input)

This field selects the data to be shifted into a register, when a Shift function is being done.

CSR<10:9> (A Adrs)

This field selects which Working Register to use as a source of data, when RAM A is selected to the ALU.

CSR<8:7> (B Adrs)

This field selects which Working Register to use as a source (when RAM B is selected to the ALU) and/or destination (when the RAM is written) of data.

CSR<6:5> (CC)

This field specifies the data type and condition code control.

CSR<4:0> (SCTL)

This field specifies the skip condition/micro sequencer function.


```

:4165 .page
:4166
:4167
:4168 4. 'MEM.REQ' Microinstruction
:4169
:4170 CSR<22:19> = 0011 (OPCODE)
:4171
:4172 This opcode causes a Memory Request to be started. The Local
:4173 Store is output on the D-Bus, to be used as the virtual
:4174 address. Working Register 5 is written with this address.
:4175
:4176 CSR<18:16>, <8:7> (Memory Function)
:4177
:4178 This field specifies to the Memory Controller the type of
:4179 request: physical, virtual, type of access, etc. For the
:4180 list of functions and the codes, see the NEBULA Memory Spec.
:4181
:4182 CSR<15:9> (D Adrs <6:0>)
:4183
:4184 This field selects which of the 128 accessible Local Store
:4185 locations to use as the virtual address.
:4186
:4187 CSR<6:5> (Data Type)
:4188
:4189 This field specifies the data type of the request.
:4190
:4191 00:=BYTE
:4192 01:=WORD
:4193 10:=SIZE Reg
:4194 11:=LONG
:4195
:4196 CSR<4:0> (SCTL)
:4197
:4198 This field specifies the skip condition/micro sequencer
:4199 function.
    
```

```

4200 page
4201
4202
4203 5. 'MISC' Microinstruction
4204
4205 CSR<22:19> = 0010 (OPCODE)
4206
4207 This combination of opcode bits causes the data path to no-op,
4208 and enables the MISC decoders.
4209
4210 CSR<18> (Port/Misc Select)
4211
4212 This bit defines the instruction to be either a Misc or a Port
4213 instruction. For CSR<18> = 1, CSR<17:10> is the Command Byte
4214 to the Port, CSR<9:5> must be zero and CSR<4:0> is the SCTL
4215 field. For CSR<18> = 0, it is a Misc instruction and the rest
4216 of the word is defined as follows.
4217
4218 CSR<17:16> (Not Used)
4219
4220 CSR<15:12> (Misc Function 1)
4221
4222 These four bits are decoded to do the following functions:
4223
4224 0 = CLEAR State Reg<1:0>
4225 1 = CLEAR State Reg<1>, SET State Reg<0>
4226 2 = SET State Reg<1>, CLEAR State Reg<0>
4227 3 = CLEAR State Reg<0>
4228 4 = CLEAR State Reg<1>
4229 5 = SET State Reg<0>
4230 6 = SET State Reg<1>
4231 7 = SET REGISTER BACKUP MASK FLAG
4232 8 = SET ACC I/O MODE
4233 9 = SET PORT I/O MODE
4234 A = CLEAR WCS HI ADRS
4235 B = SET WCS HI ADRS
4236 C = CLEAR CPU ATTN AND CPU ACK
4237 D = SET CPU ACK
4238 E = SET CPU ATTN
4239 F = NOP
4240
4241 CSR<11:9> (Misc Function 2)
4242
4243 This field is decoded to do the following functions:
4244
4245 0 = NOP
4246 1 = Mask IRD Interrupt Detection during Next State
4247 2 = Assert CPU Data Available and pass WR to ACC/Port
4248 3 = Trap ACC
4249 4 = Read ACC uPC
4250 5 = Assert XFER GRANT to Port
4251 6 = Mask Halt and T-Bit Traps
4252 7 = Write WR to Console Read Register

```

```

:4253 .page
:4254
:4255 CSR<8> (MUST BE ZERO)
:4256
:4257 CSR<7> (WR Address)
:4258
:4259
:4260 This bit selects WR[0] or WR[1] to be put on the Y-Bus.
:4261
:4262 CSR<6:5> (Not Used)
:4263
:4264 CSR<4:0> (SCTL)
:4265
:4266 This field specifies the skip condition/micro sequencer
:4267 function.
:4268

```



```

:4269 .page
:4270
:4271
:4272 6. "JUMP" Microinstruction
:4273
:4274     CSR<22:19> = 0001    (OPCODE)
:4275
:4276     This opcode causes the data path to no-op, and the micro
:4277     sequencer to execute a JUMP.
:4278
:4279     CSR<18>    (Select OS)
:4280
:4281     This bit, when asserted, causes OS<4:0> to be OR'ed with the
:4282     five low bits of the of the jump microaddress.
:4283
:4284     CSR<17:4>  (Jump Microaddress)
:4285
:4286     If Select OS is CLEAR, this field specifies the fourteen-bit
:4287     jump micro address. If Select OS is SET, CSR<17:9> is Jump
:4288     Adrs<13:5> and OS<4:0> OR'ed with CSR<8:4> is Jump Adrs<4:0>.
:4289
:4290     CSR<3:0>   (JCTL)
:4291
:4292     This field specifies the jump condition/micro sequencer
:4293     function.
    
```

:4294 .page
:4295
:4296
:4297
:4298
:4299
:4300
:4301
:4302
:4303
:4304
:4305
:4306
:4307
:4308
:4309
:4310
:4311
:4312
:4313
:4314
:4315
:4316
:4317
:4318
:4319
:4320
:4321
:4322
:4323
:4324
:4325
:4326
:4327
:4328
:4329
:4330
:4331
:4332
:4333
:4334
:4335
:4336
:4337
:4338
:4339
:4340
:4341
:4342
:4343
:4344

7. "DECODE" Microinstruction

A. DESCRIPTION

CSR<22:19> = 0000 (OPCODE)

This combination of opcode bits causes the Local Store to access location 10 (the PC.) The Local Store drives the D-Bus. The 2901 is set up to input the data on the D-Bus, increment it, and output to the Y-Bus. Thus, the Y-Bus contains the PC + 1.

CSR<18> (BPC [asserted low])

If this bit is CLEAR, the incremented PC is written into the 2901 RAM. If it is SET, the RAM is not written.

CSR<17:12> (DEC.ADRS <13:8>)

This data is taken to be the upper six bits of the next microaddress. The low eight bits are supplied by the IRD ROM. The actual JUMP/JSR/NO-OP is determined by the JCTL field.

CSR<11:9> (IFUNC)

This is the IFUNC field. It defines which fork is being taken.

CSR<8> (IB.REQ)

This is the IB.REQ bit. When it is CLEAR, the LS is not written, no interaction with memory is initiated and the instruction buffer is not affected.

When it is SET:

The incremented PC (valid on the Y-Bus by the end of the cycle) is re-written to Local Store location 0. This completes the PC increment function.

A Conditional Memory Request is executed. If the two low bits of the PC are not 11, the request is not initiated. If the two low bits are set, an IB PREFETCH is done. The (unincremented) PC is output to the memory and the CPU grant (CPUG) signal is examined. If CPUG is low by T120, the CPU will stall until the CPUG signal becomes true. After the request is completed, the next state entered is dependent upon the JCTL field.

```

:4345 :page
:4346 :
:4347 : CSR<7> (SEL CM HI IR BYTE)
:4348 :
:4349 : This bit enables the upper byte of the Compatibility Mode
:4350 : instruction in the Prefetch register to drive the IB-Bus. It
:4351 : is used only while in Compatibility Mode, when IRD is being
:4352 : done. It must never be asserted in VAX Mode.
:4353 :
:4354 : CSR<6> (LD.OS)
:4355 :
:4356 : If this bit is SET, the eight-bit data on the IB Bus (the
:4357 : output of the instruction buffer) is loaded into the OS
:4358 : register. This load is dependent upon Compatibility Mode and
:4359 : LD R Dest (If CM.AND.LD R Dest, OS<3> <- 0.) If the bit is
:4360 : CLEAR, OS is not affected.
:4361 :
:4362 : CSR<5> (LD R DEST)
:4363 :
:4364 : If this bit is SET, the R Dest Skip bit will be SET for
:4365 : Specifier Mode equal to 5 and CLEARED for Mode not equal to 5,
:4366 : in VAX mode, or mode 0 while in Compatibility Mode. If this
:4367 : bit is CLEAR, the R Dest bit will be unaffected.
:4368 :
:4369 : CSR<4> (OPC/SPEC)
:4370 :
:4371 : This bit (effectively another IFUNC bit) defines the data to
:4372 : be decoded; that is, if it is an Opcode or a Specifier. In
:4373 : hardware, it selects which ROM is to drive the micro address
:4374 : bus.
:4375 :
:4376 : CSR<3:0> (JCTL)
:4377 :
:4378 : These four bits are decoded to make one of 16 functions. See
:4379 : Table 3.
:4380 :
    
```


4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426

page "Tables"

TABLES

Table 1. 2901 Control Decoding

The 2901 control decoder examines CSR<22:14>. This nine-bit field of the microword changes in meaning for the different cases of micro opcode.

CSR	22	21	20	19	18	17	16	15	14	
1. BASIC	1	[Six bit function code]						X	X	
2. MOVE	0	1	1	[function]			X	X	X	
3. EXTENDED	0	1	0	[Six bit function code]						
4. MEM.REQ	0	0	1	1	X	X	X	X	X	WR[5] <- D
5. MISC	0	0	1	0	X	X	X	X	X	Y-Bus <- WR
6. JUMP	0	0	0	1	X	X	X	X	X	NO-OP
7. DECODE	0	0	0	0	BPC*	X	X	X	X	

If BPC = 0, Write WR; else, No-op

This decoder is implemented using a ROM and a PAL. 512 locations are needed for CSR<22:14> to index into. The decoding logic supplies 10 control bits:

ALU Source Select (3)
ALU Function (3)
ALU Result Destination (3)
ALU Carry-In (1)

The ROM addresses are therefore allocated in the following way:

Location	0 - F	Incr. D-Bus, Output ALU, No Write.
	10 - 1F	Incr. D-Bus, Output ALU, Write RAM.
	20 - 3F	No-Op.
	40 - 5F	Output WR, bypassing ALU.
	60 - 7F	Pass D-Bus through ALU and write RAM.
	80 - BF	CSR<19:14> indicates 2901 function.
	C0 - FF	CSR<19:17> indicates 2901 function.
	100 - 1FF	CSR<21:16> indicates 2901 function.

For the detailed function table see the Microcode Listing.

:4427
:4428
:4429
:4430
:4431
:4432
:4433
:4434
:4435
:4436
:4437
:4438
:4439
:4440
:4441
:4442
:4443
:4444
:4445
:4446
:4447
:4448
:4449
:4450
:4451
:4452
:4453
:4454
:4455
:4456
:4457
:4458
:4459
:4460
:4461
:4462
:4463
:4464
:4465
:4466

:page

Table 2. Local Store Write

The Local Store Write Controller examines CSR<22:17>, CSR<6>, the SIZE register (SIZE<1:0>) and the Compatibility Mode bit (CM). These eight bits are decoded to determine which parts of the LS to write, if any. The LS is written with data from the 2901 ALU (Y-Bus).

	CSR	22	21	20	19	18	17	
		--	--	--	--	--	--	
1. BASIC		1	0	X	X	X	X	No Write
		1	1	X	X	X	X	Write *
2. MOVE		0	1	1	0	X	1	No Write
		0	1	1	0	1	X	No Write
		0	1	1	1	X	X	Write *
3. EXTENDED		0	1	1	X	X	X	No Write
4. MEM.REQ		0	0	1	1	X	X	No Write
5. MISC		0	0	1	0	X	X	No Write
6. JUMP		0	0	0	1	X	X	No Write
7. DECODE		0	0	0	0	X	X	Conditional Write **

* LS Write, Data Type Dependent. These cases now examine CSR<6>.

If CSR<6> = 0, and CM = 0 Write LS<31:0> (Data Type = LONG)

If CSR<6> = 0, and CM = 1 Write LS<15:0> (Compatibility Mode. DT = WORD)

If CSR<6> = 1, the Write is conditional on the SIZE register.

If CM = 0 SIZE<1:0> = 00 Write LS<7:0> (BYTE)

= 01 Write LS<15:0> (WORD)

= 1X Write LS<31:0> (LONG)

If CM = 1 Write LS<15:0> (WORD)

** If CSR<8> = 0, No Write

If CSR<8> = 1, Write LS<31:0> (Writes incremented PC)

4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511

```

;This table summarizes the Skip and Jump conditions, and the other
;special functions of the micro sequencer.
;
;SCTL codes 0-F, 17-1F are Skip conditions. The codes 10-17 execute
;special functions. JCTL codes 0-B are Jump conditions, and C-F
;execute special functions.

```

Sctl	Function	Sctl	Function
00	Not ALU N	10	RTN+1 if No Mem Err
01	Not ALU Z	11	Loop if Not ALU Z
02	Not ALU V	12	POP Stack
03	ALU C	13	Loop if ALU C
04	Not PSL C	14	Return
05	Branch False	15	No Skip (NOP)
06	RDEST	16	RTN+1
07	Not Inter. Req.	17	Loop if No Inter. and No Sync
08	ALU N	18	Console ATTN
09	ALU Z	19	Console ACK
0A	ALU V	1A	Port Interrupt Pending
0B	Not ALU C	1B	Reg. Backup Mask Flag
0C	State 0	1C	ALU N.XOR.ALU V
0D	State 1	1D	Not Memory Error
0E	Not State 0	1E	Skip
0F	Not State 1	1F	Sync

Jctl	Function	Jctl	Function
00	Not ALU N	08	ALU N
01	Not ALU Z	09	ALU Z
02	Not ALU V	0A	ALU V
03	ALU C	0B	Not ALU C
04	JUMP	0C	JSR
05	Branch False	0D	JSR if IB Valid
06	RDEST	0E	No Jump; Skip if IB Valid
07	Not Interr. Req	0F	IB Valid

:4512

.PAGE '2901 ALU Control Description'

:4513

:4514

:4515

:4516

:4517

:4518

:4519

:4520

:4521

:4522

:4523

:4524

:4525

:4526

:4527

:4528

:4529

:4530

:4531

:4532

:4533

:4534

:4535

:4536

:4537

:4538

:4539

:4540

:4541

:4542

:4543

:4544

:4545

:4546

:4547

:4548

:4549

:4550

:4551

:4552

:4553

:4554

:4555

:4556

:4557

:4558

:4559

The 2901 ALU is the heart of the CPU module. It is responsible for all CPU calculations. The control lines going to 10 through 18 control the function, input, and output of the 2901. When a failure involving the 2901 occurs, the logical place to start looking is the control lines. The following tables describe what the control signals should be for various functions. Use it whenever the control lines are to be checked. The test descriptions tell the operator what function the 2901 is supposed to be doing.

The SRC CTL on lines 10 through 12 (pins 12 through 14 respectively) control where the inputs to the internal ALU within the 2901 come from. The inputs to the internal ALU are labeled R and S. R is one operand (4 bits) and S is the other operand (also 4 bits). The R input can come from a Working Register (labeled A), the D bus (direct input), or be forced to 0. The S input can come from either set of working registers (labeled A and B), the Q register, or forced to 0. A table of these control signals follow:

			octal code	SRC operands	
12	11	10		R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

ALU SOURCE OPERAND CONTROL

The ALU CTL on lines 13 through 15 (pins 26, 28, 27 respectively) control what function the ALU is to perform on the two operands R and S. A table follows:

:4560
:4561
:4562
:4563
:4564
:4565
:4566
:4567
:4568
:4569
:4570
:4571
:4572
:4573
:4574
:4575
:4576
:4577
:4578
:4579
:4580
:4581
:4582
:4583
:4584
:4585
:4586
:4587
:4588
:4589
:4590
:4591
:4592
:4593
:4594
:4595
:4596
:4597
:4598
:4599
:4600
:4601
:4602
:4603
:4604
:4605
:4606
:4607
:4608
:4609
:4610
:4611
:4612
:4613

page

15	14	13	octal code	ALU function
L	L	L	0	R PLUS S
L	L	H	1	S MINUS R
L	H	L	2	R MINUS S
L	H	H	3	R OR S
H	L	L	4	R AND S
H	L	H	5	R NOT AND S
H	H	L	6	R XOR S
H	H	H	7	R XNOR S

ALU FUNCTION CONTROL

The DST CTL on lines 16 through 18 (pins 5, 7, 6 respectively) control what goes out onto the Y bus (either the internal ALU result or a WR directly) and what happens internal to the 2901 (shifting, writing the Q register etc). Internal shifting left or right is accomplished with this control. Below is a table explaining the various destination possibilities. The arrow refers to where the output will end up. B indicates the WR addressed by the B ADDR, F refers to the output of the internal ALU, and A refers to the output of the WR addressed by the A ADDR. Up is toward the MSB, while down is toward the LSB.

nomenclature in listing	18	17	16	octal code	RAM function shift	load	Q-reg function shift	load	Y output
LOAD Q	L	L	L	0	NONE	NONE	NONE	F->Q	F
NOP	L	L	H	1	NONE	NONE	NONE	NONE	F
WRITE.B.A	L	H	L	2	NONE	F->B	NONE	NONE	A
WRITE.B.F	L	H	H	3	NONE	F->B	NONE	NONE	F
RSHF.RAM.Q	H	L	L	4	DOWN	F/2->B	DOWN	Q/2->Q	F
RSHF.RAM	H	L	H	5	DOWN	F/2->B	NONE	NONE	F
LSHF.RAM.Q	H	H	L	6	UP	2F->B	UP	2Q->Q	F
LSHF.RAM	H	H	H	7	UP	2F->B	NONE	NONE	F

ALU DESTINATION CONTROL

ZZ-ENKCH-1.0 ; ENKCH.MIC 2901 ALU Control De19-MAY-83 N 8
: ENKCH.MCR MICRO2 1M(01) 19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module Sequence 104
: ENKCH.MIC 2901 ALU Control Description Rev 01.00 Page 101
;4614 .page

ZZ-ENKCH-1.0
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

MICRO2 1M(01)

2901 ALU Control De19-MAY-83

2901 ALU Control Description

Fiche 1 Frame B9

Sequence 105

ENKCH Micro-diagnostic for IDC module Rev 01.00 Page 102

```
:4615 .PAGE
:4616 .TOC "DIAGNOSTIC MACROS"
:4617
:4618 LS.DATA.WORD/=<15:0>
:4619 UPPER.BYTE/=<23:16>
:4620
:4621 WORD[] "UPPER.BYTE/0,LS.DATA.WORD/@1"
:4622 COUNT.LS[] "UPPER.BYTE/0,LS.DATA.WORD/@1"
:4623 COUNT.MM[] "UPPER.BYTE/1,LS.DATA.WORD/@1"
:4624 START.ADR[] "UPPER.BYTE/0,LS.DATA.WORD/@1"
:4625
:4626
:4627
:4628 ASCII.LIT/=<15:0>
:4629
:4630 CA=4341 :ASCII VALUE FOR FILE NAMES
:4631 CB=4342 :
:4632 CC=4343 :
:4633 CD=4344 :
:4634 CE=4345 : V
:4635 CF=4346
:4636 CG=4347
:4637 CH=4348
:4638
:4639
:4640 ASCII [] "UPPER.BYTE/0,ASCII.LIT/@1"
:4641
:4642
:4643 .REGION/0,6F
```

:4644 .PAGE "TEST POINTERS"

:4645
 :4646 This portion contains the pointers to all tests in this section. The
 :4647 WCS address of the JUMP instruction corresponds to the test number.
 :4648 E.g. WCS 5 contains a JUMP to test 5. All unused test numbers will
 :4649 contain a JUMP to an error routine. This portion is 80. locations
 :4650 long, allowing for up to 80. tests per section, from WCS address 1 to
 :4651 WCS address 4F.
 :4652
 :4653
 :4654
 :4655
 :4656

:TEST POINTERS BEGIN AT WCS ADDRESS 1

U 001, 0870,04	:4657	JMP [T.1]	:GO TO TEST 1
U 002, 886B,E4	:4658	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4659		:IN THIS OVERLAY
U 003, 886B,E4	:4660	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4661		:IN THIS OVERLAY
U 004, 886B,E4	:4662	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4663		:IN THIS OVERLAY
U 005, 886B,E4	:4664	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4665		:IN THIS OVERLAY
U 006, 886B,E4	:4666	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4667		:IN THIS OVERLAY
U 007, 886B,E4	:4668	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4669		:IN THIS OVERLAY
U 008, 886B,E4	:4670	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4671		:IN THIS OVERLAY
U 009, 886B,E4	:4672	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4673		:IN THIS OVERLAY
U 00A, 886B,E4	:4674	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4675		:IN THIS OVERLAY
U 00B, 886B,E4	:4676	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4677		:IN THIS OVERLAY
U 00C, 886B,E4	:4678	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4679		:IN THIS OVERLAY
U 00D, 886B,E4	:4680	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4681		:IN THIS OVERLAY
U 00E, 886B,E4	:4682	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4683		:IN THIS OVERLAY
U 00F, 886B,E4	:4684	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4685		:IN THIS OVERLAY
U 010, 886B,E4	:4686	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4687		:IN THIS OVERLAY
U 011, 886B,E4	:4688	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4689		:IN THIS OVERLAY
U 012, 886B,E4	:4690	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4691		:IN THIS OVERLAY
U 013, 886B,E4	:4692	JMP [ERR.NO.TEST]	:GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
	:4693		:IN THIS OVERLAY

```

:4694 .PAGE "DIAGNOSTIC POINTERS"
:4695
:4696
:4697 This section contains all the pointers needed for this diagnostic
:4698 overlay. The first pointer (at WCS location 0) points to a data
:4699 transfer routine. It is the first location executed after a new
:4700 WCS load. The instruction here is a JUMP to TRANSFER.POINT which may
:4701 contain instructions to transfer data. If no data is to be trans-
:4702 ferred, or at the completion of the data transfers, the CPU will
:4703 issue CPU ATTN with all bits of the control and status word clear.
:4704
:4705 The next 80. locations (from WCS location 1 to WCS location 4F)
:4706 contain pointers to each test in this overlay. The pointers are
:4707 JUMP instructions to the test number corresponding to the location
:4708 number. E.g. location 5 will contain a JUMP to test 5. All unused
:4709 test numbers will contain a JUMP to an error routine. This portion
:4710 appears after the definitions in this listing.
:4711
:4712 Finally the next 32. locations (from WCS location 50 to 6F) contain
:4713 pointers (Jump instructions) to WCS subroutines which are to be used by
:4714 the console diagnostic monitor.
:4715
U 000, 086C,24 :4716 0: JMP [TRANSFER.POINT] ;GO TO ROUTINE THAT LOADS LS 7 WITH
:4717 ; POINTER TO DATA SECTION AND ISSUES
:4718 ; CPU ATTN WITH TRANSFER BIT SET.
:4719
:4720 50: ;START AT WCS LOCATION 50 FOR SUB-
:4721 ; ROUTINE POINTERS
:4722
:4723 SUBROUTINE POINTERS
:4724
U 050, 0860,84 :4725 JMP [DEPOSIT.CSR] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4726 ; MCT CSR FOR DEPOSIT CSR COMMAND
U 051, 0860,D4 :4727 JMP [EXAMINE.CSR] ;GO TO WCS SUBROUTINE WHICH READS THE
:4728 ; MCT CSR FOR EXAMINE CSR COMMAND
U 052, 0861,F4 :4729 JMP [DEPOSIT.TB] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4730 ; MCT TRANSLATION BUFFER FOR DEPOSIT
:4731 ; TB COMMAND
U 053, 8863,A4 :4732 JMP [EXAMINE.TB] ;GO TO WCS SUBROUTINE WHICH READS THE
:4733 ; MCT TRANSLATION BUFFER FOR EXAMINE
:4734 ; TB COMMAND
U 054, 8865,C4 :4735 JMP [DEPOSIT.MM] ;GO TO WCS SUBROUTINE WHICH WRITES MAIN
:4736 ; MEMORY FOR DEPOSIT MM COMMAND. ALSO
:4737 ; USED TRANSFERING DATA FROM WCS TO
:4738 ; MAIN MEMORY.
U 055, 0866,24 :4739 JMP [EXAMINE.MM] ;GO TO WCS SUBROUTINE WHICH READS MAIN
:4740 ; MEMORY FOR EXAMINE MM COMMAND.
U 056, 0868,94 :4741 JMP [SAVE.WR] ;GO TO WCS SUBROUTINE WHICH STORES THE
:4742 ; CONTENTS OF WR0-WR3 IN LS 0-3
U 057, 8868,E4 :4743 JMP [RESTORE.WR] ;GO TO WCS SUBROUTINE WHICH RESTORES THE
:4744 ; CONTENTS OF WR0-WR3 FROM LS 0-3
U 058, 8864,24 :4745 JMP [DEPOSIT.UBS] ;GO TO WCS SUBROUTINE WHICH WRITES THE
:4746 ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 059, 0865,44 :4747 JMP [EXAMINE.UBS] ;GO TO WCS SUBROUTINE WHICH READS THE
:4748 ; UNIBUS MAP ON THE MEMORY CONTROLLER

```


U 05A, 0866,84	:4749	JMP [EXAMINE.ICSR]	:GO TO WCS SUBROUTINE WHICH READS THE
	:4750		:IDC CSR
U 05B, 0866,B4	:4751	JMP [EXAMINE.IDAR]	:GO TO WCS SUBROUTINE WHICH READS THE
	:4752		:IDC DAR
U 05C, 0866,E4	:4753	JMP [EXAMINE.DBUF]	:GO TO WCS SUBROUTINE WHICH READS THE
	:4754		:IDC DBUF
U 05D, 8867,14	:4755	JMP [EXAMINE.PATT]	:GO TO WCS SUBROUTINE WHICH READS THE
	:4756		:IDC ECC PATTERN
U 05E, 8867,44	:4757	JMP [EXAMINE.POSIT]	:GO TO WCS SUBROUTINE WHICH READS THE
	:4758		:IDC ECC POSITION
U 05F, 0867,94	:4759	JMP [DEPOSIT.ICSR]	:GO TO WCS SUBROUTINE WHICH WRITES THE
	:4760		:IDC CSR
U 060, 0867,C4	:4761	JMP [DEPOSIT.IDAR]	:GO TO WCS SUBROUTINE WHICH WRITES THE
	:4762		:IDC DAR
U 061, 0867,F4	:4763	JMP [DEPOSIT.DBUF]	:GO TO WCS SUBROUTINE WHICH WRITES THE
	:4764		:IDC DBUF
U 062, 8868,24	:4765	JMP [CLEAR.FIFO.ADDR]	:GO TO WCS SUBROUTINE WHICH CLEARS
	:4766		:THE IDC FIFO ADDRESS
U 063, 8868,44	:4767	JMP [SELECT.FIFO.A]	:GO TO WCS SUBROUTINE WHICH SELECTS
	:4768		:FIFO A
U 064, 0868,64	:4769	JMP [SELECT.FIFO.B]	:GO TO WCS SUBROUTINE WHICH SELECTS
	:4770		:FIFO B
U 065, DB00,15	:4771	NOP	:NOT USED
U 066, DB00,15	:4772	NOP	:NOT USED
U 067, DB00,15	:4773	NOP	:NOT USED
U 068, DB00,15	:4774	NOP	:NOT USED
U 069, DB00,15	:4775	NOP	:NOT USED
U 06A, DB00,15	:4776	NOP	:NOT USED
U 06B, DB00,15	:4777	NOP	:NOT USED
U 06C, DB00,15	:4778	NOP	:NOT USED
U 06D, DB00,15	:4779	NOP	:NOT USED
U 06E, DB00,15	:4780	NOP	:NOT USED
U 06F, DB00,15	:4781	NOP	:NOT USED

:4782 .PAGE
 :4783 .REGION/70,5FF
 :4784 .TOC 'LS DATA SECTION'
 :4785

:4786 :+++

This section lists the data that will be transferred to LS by the console as part of the initialization performed in test 0. The TRANSFER DATA routine will point to this data area. LS contains the constants, control and status word, and temporary storage locations used by the tests and subroutines of the microdiagnostic.

The LS is 32 bits wide. Each of these words is 16 bits, so two consecutive words listed here will be loaded into each consecutive LS location. The first word listed will be bits 0-15 of the LS location, the second word listed will be bits 16-31 of the same LS location.

The locations 78-7F in the first half of LS and F8-FF in the second half of LS are not actual LS locations. They force an access to one of several registers in the CPU or force an indexing feature to another LS location. For that reason, they are not written via the transfer routine.

Note: This table is in hexadecimal format i.e. each digit represents four bits, so four digits represents a full 16 bit word. The micro-assembler requires that a hexadecimal value that starts with a letter (e.g. FFFF) be preceded by a 0 (0FFFF). So some table values will contain 5 hexadecimal digits. The fifth digit is not actually used.

:4807 :---

TRANSFER.DATA.LS1:

U 070, 0000,78	COUNT.LS[0078]	:LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANSFER TO LS = 120 DECIMAL) DESTINATION IS LS
U 071, 0000,00	START.ADR[0000]	:DESTINATION START ADDRESS (START AT LS 0)
U 072, 0000,00	WORD[0000]	:LOCATION 0
U 073, 0000,00	WORD[0000]	
U 074, 0000,00	WORD[0000]	:LOCATION 1
U 075, 0000,00	WORD[0000]	
U 076, 0000,00	WORD[0000]	:LOCATION 2
U 077, 0000,00	WORD[0000]	
U 078, 0000,00	WORD[0000]	:LOCATION 3
U 079, 0000,00	WORD[0000]	
U 07A, 0000,00	WORD[0000]	:LOCATION 4
U 07B, 0000,00	WORD[0000]	
U 07C, 0000,00	WORD[0000]	:LOCATION 5
U 07D, 0000,00	WORD[0000]	
U 07E, 0000,00	WORD[0000]	:LOCATION 6
U 07F, 0000,00	WORD[0000]	
U 080, 0000,00	WORD[0000]	:LOCATION 7

ZZ-ENKCH-1.0	:	ENKCH.MIC	LS DATA SECTION	6 9	19-MAY-83	13:28:52	Fiche 1 Frame G9	Sequence 110	Page 107
:	:	ENKCH.MCR	MICRO2 1M(01)				ENKCH Micro-diagnostic for IDC module	Rev 01.00	
:	:	ENKCH.MIC	LS DATA SECTION						

U 081, 0000,00	:4837	WORD[0000]	
	:4838		
U 082, 0000,00	:4839	WORD[0000]	:LOCATION 8
U 083, 0000,00	:4840	WORD[0000]	
	:4841		
U 084, 0000,00	:4842	WORD[0000]	:LOCATION 9
U 085, 0000,00	:4843	WORD[0000]	
	:4844		
U 086, 0000,00	:4845	WORD[0000]	:LOCATION 0A
U 087, 0000,00	:4846	WORD[0000]	
	:4847		
U 088, 0000,00	:4848	WORD[0000]	:LOCATION 0B
U 089, 0000,00	:4849	WORD[0000]	
	:4850		
U 08A, 0000,00	:4851	WORD[0000]	:LOCATION 0C
U 08B, 0000,00	:4852	WORD[0000]	
	:4853		
U 08C, 0000,00	:4854	WORD[0000]	:LOCATION 0D
U 08D, 0000,00	:4855	WORD[0000]	
	:4856		
U 08E, 0000,00	:4857	WORD[0000]	:LOCATION 0E
U 08F, 0000,00	:4858	WORD[0000]	
	:4859		
U 090, 0000,00	:4860	WORD[0000]	:LOCATION 0F
U 091, 0000,00	:4861	WORD[0000]	
	:4862		
U 092, 0000,00	:4863	WORD[0000]	:LOCATION 10
U 093, 0000,00	:4864	WORD[0000]	
	:4865		
U 094, 0000,00	:4866	WORD[0000]	:LOCATION 11
U 095, 0000,00	:4867	WORD[0000]	
	:4868		
U 096, 0000,00	:4869	WORD[0000]	:LOCATION 12
U 097, 0000,00	:4870	WORD[0000]	
	:4871		
U 098, 0000,FF	:4872	WORD[00FF]	:LOCATION 13
U 099, 0000,00	:4873	WORD[0000]	
	:4874		
U 09A, 00FF,FF	:4875	WORD[0FFFF]	:LOCATION 14
U 09B, 0000,00	:4876	WORD[0000]	
	:4877		
U 09C, 0000,00	:4878	WORD[0000]	:LOCATION 15
U 09D, 00FF,00	:4879	WORD[0FF00]	
	:4880		
U 09E, 00FF,00	:4881	WORD[0FF00]	:LOCATION 16
U 09F, 00FF,FF	:4882	WORD[0FFFF]	
	:4883		
U 0A0, 003F,FF	:4884	WORD[3FFF]	:LOCATION 17
U 0A1, 0001,00	:4885	WORD[0100]	
	:4886		
U 0A2, 003F,FF	:4887	WORD[3FFF]	:LOCATION 18
U 0A3, 007F,FE	:4888	WORD[7FFE]	
	:4889		
U 0A4, 00FF,FF	:4890	WORD[0FFFF]	:LOCATION 19
U 0A5, 00FE,7F	:4891	WORD[0FE7F]	

ZZ
...

22
22
22

22

U 0A6, 0000.00	:4892		
U 0A7, 007F.F8	:4893	WORD[0000]	:LOCATION 1A
	:4894	WORD[7FF8]	
	:4895		
U 0A8, 0080.00	:4896	WORD[8000]	:LOCATION 1B
U 0A9, 007F.FF	:4897	WORD[7FFF]	
	:4898		
U 0AA, 0000.00	:4899	WORD[0000]	:LOCATION 1C
U 0AB, 0000.00	:4900	WORD[0000]	
	:4901		
U 0AC, 0000.00	:4902	WORD[0000]	:LOCATION 1D
U 0AD, 0000.00	:4903	WORD[0000]	
	:4904		
U 0AE, 0005.00	:4905	WORD[0500]	:LOCATION 1E
U 0AF, 0000.80	:4906	WORD[0080]	
	:4907		
U 0B0, 0005.00	:4908	WORD[0500]	:LOCATION 1F
U 0B1, 0000.00	:4909	WORD[0000]	
	:4910		
U 0B2, 0000.01	:4911	WORD[0001]	:LOCATION 20
U 0B3, 0000.00	:4912	WORD[0000]	
	:4913		
U 0B4, 0000.02	:4914	WORD[0002]	:LOCATION 21
U 0B5, 0000.00	:4915	WORD[0000]	
	:4916		
U 0B6, 0000.04	:4917	WORD[0004]	:LOCATION 22
U 0B7, 0000.00	:4918	WORD[0000]	
	:4919		
U 0B8, 0000.08	:4920	WORD[0008]	:LOCATION 23
U 0B9, 0000.00	:4921	WORD[0000]	
	:4922		
U 0BA, 0000.10	:4923	WORD[0010]	:LOCATION 24
U 0BB, 0000.00	:4924	WORD[0000]	
	:4925		
U 0BC, 0000.20	:4926	WORD[0020]	:LOCATION 25
U 0BD, 0000.00	:4927	WORD[0000]	
	:4928		
U 0BE, 0000.40	:4929	WORD[0040]	:LOCATION 26
U 0BF, 0000.00	:4930	WORD[0000]	
	:4931		
U 0C0, 0000.80	:4932	WORD[0080]	:LOCATION 27
U 0C1, 0000.00	:4933	WORD[0000]	
	:4934		
U 0C2, 0001.00	:4935	WORD[0100]	:LOCATION 28
U 0C3, 0000.00	:4936	WORD[0000]	
	:4937		
U 0C4, 0002.00	:4938	WORD[0200]	:LOCATION 29
U 0C5, 0000.00	:4939	WORD[0000]	
	:4940		
U 0C6, 0004.00	:4941	WORD[0400]	:LOCATION 2A
U 0C7, 0000.00	:4942	WORD[0000]	
	:4943		
U 0C8, 0008.00	:4944	WORD[0800]	:LOCATION 2B
U 0C9, 0000.00	:4945	WORD[0000]	
	:4946		

ZZ-ENKCH-1.0	:	ENKCH.MIC	LS DATA SECTION	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00	Page 109
:	:	ENKCH.MCR	MICRO2 1M(01)					
:	:	ENKCH.MIC	LS DATA SECTION					

U OCA, 0010.00	:4947	WORD[1000]	:LOCATION 2C
U OCB, 0000.00	:4948	WORD[00C0]	
	:4949		
U OCC, 0020.00	:4950	WORD[2000]	:LOCATION 2D
U OCD, 0000.00	:4951	WORD[0000]	
	:4952		
U OCE, 0040.00	:4953	WORD[4000]	:LOCATION 2E
U OCF, 0000.00	:4954	WORD[0000]	
	:4955		
U ODO, 0080.00	:4956	WORD[8000]	:LOCATION 2F
U OD1, 0000.00	:4957	WORD[0000]	
	:4958		
U OD2, 0000.00	:4959	WORD[0000]	:LOCATION 30
U OD3, 0000.01	:4960	WORD[0001]	
	:4961		
U OD4, 0000.00	:4962	WORD[0000]	:LOCATION 31
U OD5, 0000.02	:4963	WORD[0002]	
	:4964		
U OD6, 0000.00	:4965	WORD[0000]	:LOCATION 32
U OD7, 0000.04	:4966	WORD[0004]	
	:4967		
U OD8, 0000.00	:4968	WORD[0000]	:LOCATION 33
U OD9, 0000.08	:4969	WORD[0008]	
	:4970		
U ODA, 0000.00	:4971	WORD[0000]	:LOCATION 34
U ODB, 0000.10	:4972	WORD[0010]	
	:4973		
U ODC, 0000.00	:4974	WORD[0000]	:LOCATION 35
U ODD, 0000.20	:4975	WORD[0020]	
	:4976		
U ODE, 0000.00	:4977	WORD[0000]	:LOCATION 36
U ODF, 0000.40	:4978	WORD[0040]	
	:4979		
U OEO, 0000.00	:4980	WORD[0000]	:LOCATION 37
U OE1, 0000.80	:4981	WORD[0080]	
	:4982		
U OE2, 0000.00	:4983	WORD[0000]	:LOCATION 38
U OE3, 0001.00	:4984	WORD[0100]	
	:4985		
U OE4, 0000.00	:4986	WORD[0000]	:LOCATION 39
U OE5, 0002.00	:4987	WORD[0200]	
	:4988		
U OE6, 0000.00	:4989	WORD[0000]	:LOCATION 3A
U OE7, 0004.00	:4990	WORD[0400]	
	:4991		
U OE8, 0000.00	:4992	WORD[0000]	:LOCATION 3B
U OE9, 0008.00	:4993	WORD[0800]	
	:4994		
U OEA, 0000.00	:4995	WORD[0000]	:LOCATION 3C
U OEB, 0010.00	:4996	WORD[1000]	
	:4997		
U OEC, 0000.00	:4998	WORD[0000]	:LOCATION 3D
U OED, 0020.00	:4999	WORD[2000]	
	:5000		
U OEE, 0000.00	:5001	WORD[0000]	:LOCATION 3E

ZZ-ENKCH-1.0	:	ENKCH.MIC	LS DATA SECTION	J ⁹ 19-MAY-83	Fiche 1 Frame J9	Sequence 113	Page 110
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83 13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00	
:	:	ENKCH.MIC	LS DATA SECTION				

U 0EF, 0040,00	:5002	WORD[4000]	
	:5003		
U 0F0, 0000,00	:5004	WORD[0000]	:LOCATION 3F
U 0F1, 0080,00	:5005	WORD[8000]	
	:5006		
U 0F2, 0000,00	:5007	WORD[0000]	:LOCATION 40
U 0F3, 0000,00	:5008	WORD[0000]	
	:5009		
U 0F4, 0000,00	:5010	WORD[0000]	:LOCATION 41
U 0F5, 0000,00	:5011	WORD[0000]	
	:5012		
U 0F6, 0000,00	:5013	WORD[0000]	:LOCATION 42
U 0F7, 0000,00	:5014	WORD[0000]	
	:5015		
U 0F8, 0000,00	:5016	WORD[0000]	:LOCATION 43
U 0F9, 0000,00	:5017	WORD[0000]	
	:5018		
U 0FA, 0000,00	:5019	WORD[0000]	:LOCATION 44
U 0FB, 0000,00	:5020	WORD[0000]	
	:5021		
U 0FC, 0000,00	:5022	WORD[0000]	:LOCATION 45
U 0FD, 0000,00	:5023	WORD[0000]	
	:5024		
U 0FE, 0000,00	:5025	WORD[0000]	:LOCATION 46
U 0FF, 0000,00	:5026	WORD[0000]	
	:5027		
U 100, 0000,00	:5028	WORD[0000]	:LOCATION 47
U 101, 0000,00	:5029	WORD[0000]	
	:5030		
U 102, 0000,00	:5031	WORD[0000]	:LOCATION 48
U 103, 0000,00	:5032	WORD[0000]	
	:5033		
U 104, 0000,00	:5034	WORD[0000]	:LOCATION 49
U 105, 0000,00	:5035	WORD[0000]	
	:5036		
U 106, 0000,00	:5037	WORD[0000]	:LOCATION 4A
U 107, 0000,00	:5038	WORD[0000]	
	:5039		
U 108, 0000,00	:5040	WORD[0000]	:LOCATION 4B
U 109, 0000,00	:5041	WORD[0000]	
	:5042		
U 10A, 00AA,AA	:5043	WORD[0AAAA]	:LOCATION 4C
U 10B, 00AA,AA	:5044	WORD[0AAAA]	
	:5045		
U 10C, 0055,55	:5046	WORD[5555]	:LOCATION 4D
U 10D, 0055,55	:5047	WORD[5555]	
	:5048		
U 10E, 0000,00	:5049	WORD[0000]	:LOCATION 4E
U 10F, 0000,00	:5050	WORD[0000]	
	:5051		
U 110, 00FF,FF	:5052	WORD[0FFFF]	:LOCATION 4F
U 111, 00FF,FF	:5053	WORD[0FFFF]	
	:5054		
U 112, 0000,00	:5055	WORD[0000]	:LOCATION 50
U 113, 0000,00	:5056	WORD[0000]	

Address	Value	Label	Location
U 114,	0000,00	WORD[0000]	;LOCATION 51
U 115,	0000,00	WORD[0000]	
U 116,	0000,00	WORD[0000]	;LOCATION 52
U 117,	0000,00	WORD[0000]	
U 118,	0000,00	WORD[0000]	;LOCATION 53
U 119,	0000,00	WORD[0000]	
U 11A,	0000,00	WORD[0000]	;LOCATION 54
U 11B,	0000,00	WORD[0000]	
U 11C,	0000,00	WORD[0000]	;LOCATION 55
U 11D,	0000,00	WORD[0000]	
U 11E,	0000,00	WORD[0000]	;LOCATION 56
U 11F,	0000,00	WORD[0000]	
U 120,	0000,00	WORD[0000]	;LOCATION 57
U 121,	0000,00	WORD[0000]	
U 122,	0000,00	WORD[0000]	;LOCATION 58
U 123,	0000,00	WORD[0000]	
U 124,	0000,00	WORD[0000]	;LOCATION 59
U 125,	0000,00	WORD[0000]	
U 126,	00F0,00	WORD[0F000]	;LOCATION 5A
U 127,	00FF,FF	WORD[0FFFF]	
U 128,	00F0,02	WORD[0F002]	;LOCATION 5B
U 129,	00FF,FF	WORD[0FFFF]	
U 12A,	00F0,04	WORD[0F004]	;LOCATION 5C
U 12B,	00FF,FF	WORD[0FFFF]	
U 12C,	00F0,06	WORD[0F006]	;LOCATION 5D
U 12D,	00FF,FF	WORD[0FFFF]	
U 12E,	00F0,08	WORD[0F008]	;LOCATION 5E
U 12F,	00FF,FF	WORD[0FFFF]	
U 130,	00F0,0E	WORD[0F00E]	;LOCATION 5F
U 131,	00FF,FF	WORD[0FFFF]	
U 132,	0000,03	WORD[0003]	;LOCATION 60
U 133,	0000,00	WORD[0000]	
U 134,	0000,12	WORD[0012]	;LOCATION 61
U 135,	0000,00	WORD[0000]	
U 136,	0033,33	WORD[3333]	;LOCATION 62
U 137,	0033,33	WORD[3333]	

ZZ-ENKCH-1.0		ENKCH.MIC	LS DATA SECTION		19-MAY-83 13:28:52		Fiche 1 Frame L9		Sequence 115		Page 112
:	ENKCH.MCR	MICRO2 1M(01)	LS DATA SECTION								
:	ENKCH.MIC										
U 138,	000F,OF	:5112	WORD[0F0F]		:LOCATION 63						
U 139,	000F,OF	:5113	WORD[0F0F]								
		:5114									
U 13A,	0000,FF	:5115	WORD[00FF]		:LOCATION 64						
U 13B,	0000,FF	:5116	WORD[00FF]								
		:5117									
U 13C,	0001,62	:5118	WORD[0162]		:LOCATION 65						
U 13D,	0000,00	:5119	WORD[0000]								
		:5120									
U 13E,	0000,28	:5121	WORD[0028]		:LOCATION 66						
U 13F,	0000,00	:5122	WORD[0000]								
		:5123									
U 140,	0000,0C	:5124	WORD[000C]		:LOCATION 67						
U 141,	0000,00	:5125	WORD[0000]								
		:5126									
U 142,	0000,00	:5127	WORD[0000]		:LOCATION 68						
U 143,	0000,00	:5128	WORD[0000]								
		:5129									
U 144,	0000,00	:5130	WORD[0000]		:LOCATION 69						
U 145,	0000,00	:5131	WORD[0000]								
		:5132									
U 146,	0000,00	:5133	WORD[0000]		:LOCATION 6A						
U 147,	0000,00	:5134	WORD[0000]								
		:5135									
U 148,	0000,00	:5136	WORD[0000]		:LOCATION 6B						
U 149,	0000,00	:5137	WORD[0000]								
		:5138									
U 14A,	0000,00	:5139	WORD[0000]		:LOCATION 6C						
U 14B,	0000,00	:5140	WORD[0000]								
		:5141									
U 14C,	0000,00	:5142	WORD[0000]		:LOCATION 6D						
U 14D,	0000,00	:5143	WORD[0000]								
		:5144									
U 14E,	0000,00	:5145	WORD[0000]		:LOCATION 6E						
U 14F,	0000,00	:5146	WORD[0000]								
		:5147									
U 150,	0000,00	:5148	WORD[0000]		:LOCATION 6F						
U 151,	0000,00	:5149	WORD[0000]								
		:5150									
U 152,	0000,00	:5151	WORD[0000]		:LOCATION 70						
U 153,	0000,00	:5152	WORD[0000]								
		:5153									
U 154,	0000,00	:5154	WORD[0000]		:LOCATION 71						
U 155,	0000,00	:5155	WORD[0000]								
		:5156									
U 156,	0000,00	:5157	WORD[0000]		:LOCATION 72						
U 157,	0000,00	:5158	WORD[0000]								
		:5159									
U 158,	0000,00	:5160	WORD[0000]		:LOCATION 73						
U 159,	0000,00	:5161	WORD[0000]								
		:5162									
U 15A,	0000,00	:5163	WORD[0000]		:LOCATION 74						
U 15B,	0000,00	:5164	WORD[0000]								
		:5165									
U 15C,	0000,00	:5166	WORD[0000]		:LOCATION 75						

ZZ-ENKCH-1.0	:	ENKCH.MIC	LS DATA SECTION	M 9	19-MAY-83	Fiche 1 Frame M9	Sequence 116	Page 113
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00	
:	:	ENKCH.MIC	LS DATA SECTION					

U 15D, 0000,00	:5167	WORD[0000]	
	:5168		
U 15E, 0000,00	:5169	WORD[0000]	:LOCATION 76
U 15F, 0000,00	:5170	WORD[0000]	
	:5171		
U 160, 0000,00	:5172	WORD[0000]	:LOCATION 77
U 161, 0000,00	:5173	WORD[0000]	
	:5174		
	:5175		
	:5176		
	:5177		
	:5178		
	:5179		
	:5180		
	:5181		
U 162, 0000,78	:5182	TRANSFER.DATA.LS2:	
	:5183	COUNT.LS[0078]	:LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
U 163, 0000,80	:5184	START.ADR[0080]	:FER TO LS = 120 DECIMAL) DESTINATION IS LS
	:5185		:DESTINATION START ADDRESS (START AT LS 80(H))
U 164, 0000,00	:5186	WORD[0000]	:LOCATION 80
U 165, 0010,00	:5187	WORD[1000]	
	:5188		
U 166, 0000,02	:5189	WORD[0002]	:LOCATION 81
U 167, 0010,00	:5190	WORD[1000]	
	:5191		
U 168, 0000,04	:5192	WORD[0004]	:LOCATION 82
U 169, 0010,00	:5193	WORD[1000]	
	:5194		
U 16A, 0000,06	:5195	WORD[0006]	:LOCATION 83
U 16B, 0010,00	:5196	WORD[1000]	
	:5197		
U 16C, 0000,08	:5198	WORD[0008]	:LOCATION 84
U 16D, 0010,00	:5199	WORD[1000]	
	:5200		
U 16E, 0000,0A	:5201	WORD[000A]	:LOCATION 85
U 16F, 0010,00	:5202	WORD[1000]	
	:5203		
U 170, 0000,0C	:5204	WORD[000C]	:LOCATION 86
U 171, 0010,00	:5205	WORD[1000]	
	:5206		
U 172, 0000,0E	:5207	WORD[000E]	:LOCATION 87
U 173, 0010,00	:5208	WORD[1000]	
	:5209		
U 174, 0000,40	:5210	WORD[0040]	:LOCATION 88
U 175, 0000,00	:5211	WORD[0000]	
	:5212		
U 176, 0000,80	:5213	WORD[0080]	:LOCATION 89
U 177, 0000,00	:5214	WORD[0000]	
	:5215		
U 178, 0000,00	:5216	WORD[0000]	:LOCATION 8A
U 179, 0000,00	:5217	WORD[0000]	
	:5218		
U 17A, 0001,00	:5219	WORD[0100]	:LOCATION 8B
U 17B, 0000,00	:5220	WORD[0000]	
	:5221		

ZZ-ENKCH-1.0	:	ENKCH.MIC	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)	N 9	19-MAY-83 13:28:52	Fiche 1 Frame N9	Sequence 117	Page 114
:	:	ENKCH.MCR	MICRO2 1M(01)			ENKCH Micro-diagnostic for IDC module	Rev 01.00	
:	:	ENKCH.MIC						

U 17C, 0002.00	:5222	WORD[0200]	:LOCATION 8C
U 17D, 0000.00	:5223	WORD[0000]	
	:5224		
U 17E, 0003.00	:5225	WORD[0300]	:LOCATION 8D
U 17F, 0000.00	:5226	WORD[0000]	
	:5227		
U 180, 0000.00	:5228	WORD[0000]	:LOCATION 8E
U 181, 0000.40	:5229	WORD[0040]	
	:5230		
U 182, 0000.00	:5231	WORD[0000]	:LOCATION 8F
U 183, 0000.80	:5232	WORD[0080]	
	:5233		
U 184, 0000.00	:5234	WORD[0000]	:LOCATION 90
U 185, 0002.00	:5235	WORD[0200]	
	:5236		
U 186, 00FC.31	:5237	WORD[0FC31]	:LOCATION 91
U 187, 00C5.3F	:5238	WORD[0C53F]	
	:5239		
U 188, 0000.00	:5240	WORD[0000]	:LOCATION 92
U 189, 0000.00	:5241	WORD[0000]	
	:5242		
U 18A, 0001.44	:5243	WORD[0144]	:LOCATION 93
U 18B, 0002.40	:5244	WORD[0240]	
	:5245		
U 18C, 0002.8A	:5246	WORD[028A]	:LOCATION 94
U 18D, 0000.80	:5247	WORD[0080]	
	:5248		
U 18E, 0001.86	:5249	WORD[0186]	:LOCATION 95
U 18F, 0002.80	:5250	WORD[0280]	
	:5251		
U 190, 0000.4E	:5252	WORD[004E]	:LOCATION 96
U 191, 0002.80	:5253	WORD[0280]	
	:5254		
U 192, 0003.CE	:5255	WORD[03CE]	:LOCATION 97
U 193, 0000.40	:5256	WORD[0040]	
	:5257		
U 194, 0000.1A	:5258	WORD[001A]	:LOCATION 98
U 195, 0000.00	:5259	WORD[0000]	
	:5260		
U 196, 0000.00	:5261	WORD[0000]	:LOCATION 99
U 197, 0000.00	:5262	WORD[0000]	
	:5263		
U 198, 0000.00	:5264	WORD[0000]	:LOCATION 9A
U 199, 0000.00	:5265	WORD[0000]	
	:5266		
U 19A, 0000.00	:5267	WORD[0000]	:LOCATION 9B
U 19B, 0000.00	:5268	WORD[0000]	
	:5269		
U 19C, 0000.00	:5270	WORD[0000]	:LOCATION 9C
U 19D, 0000.00	:5271	WORD[0000]	
	:5272		
U 19E, 0000.00	:5273	WORD[0000]	:LOCATION 9D
U 19F, 0000.00	:5274	WORD[0000]	
	:5275		
U 1A0, 0000.00	:5276	WORD[0000]	:LOCATION 9E

ZZ-ENKCH-1.0		B 10		Fiche 1 Frame B10		Sequence 118	
:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83 13:28:52	ENKCH Micro-diagnostic for IDC module		Rev 01.00	Page 115
:	ENKCH.MIC	PROGRAM SPECIFIC DATA SECTION (LS 80-F7)					

U 1A1, 0000.00	:5277	WORD[0000]	
	:5278		
U 1A2, 0000.00	:5279	WORD[0000]	:LOCATION 9F
U 1A3, 0000.00	:5280	WORD[0000]	
	:5281		
U 1A4, 0000.00	:5282	WORD[0000]	:LOCATION 0A0
U 1A5, 0000.00	:5283	WORD[0000]	
	:5284		
U 1A6, 0000.00	:5285	WORD[0000]	:LOCATION 0A1
U 1A7, 0000.00	:5286	WORD[0000]	
	:5287		
U 1A8, 0000.00	:5288	WORD[0000]	:LOCATION 0A2
U 1A9, 0000.00	:5289	WORD[0000]	
	:5290		
U 1AA, 0000.00	:5291	WORD[0000]	:LOCATION 0A3
U 1AB, 0000.00	:5292	WORD[0000]	
	:5293		
U 1AC, 0000.00	:5294	WORD[0000]	:LOCATION 0A4
U 1AD, 0000.00	:5295	WORD[0000]	
	:5296		
U 1AE, 0000.00	:5297	WORD[0000]	:LOCATION 0A5
U 1AF, 0000.00	:5298	WORD[0000]	
	:5299		
U 1B0, 0000.00	:5300	WORD[0000]	:LOCATION 0A6
U 1B1, 0000.00	:5301	WORD[0000]	
	:5302		
U 1B2, 0000.00	:5303	WORD[0000]	:LOCATION 0A7
U 1B3, 0000.00	:5304	WORD[0000]	
	:5305		
U 1B4, 0000.00	:5306	WORD[0000]	:LOCATION 0A8
U 1B5, 0000.00	:5307	WORD[0000]	
	:5308		
U 1B6, 0000.00	:5309	WORD[0000]	:LOCATION 0A9
U 1B7, 0000.00	:5310	WORD[0000]	
	:5311		
U 1B8, 0000.00	:5312	WORD[0000]	:LOCATION 0AA
U 1B9, 0000.00	:5313	WORD[0000]	
	:5314		
U 1BA, 0000.00	:5315	WORD[0000]	:LOCATION 0AB
U 1BB, 0000.00	:5316	WORD[0000]	
	:5317		
U 1BC, 0000.00	:5318	WORD[0000]	:LOCATION 0AC
U 1BD, 0000.00	:5319	WORD[0000]	
	:5320		
U 1BE, 0000.00	:5321	WORD[0000]	:LOCATION 0AD
U 1BF, 0000.00	:5322	WORD[0000]	
	:5323		
U 1C0, 0000.00	:5324	WORD[0000]	:LOCATION 0AE
U 1C1, 0000.00	:5325	WORD[0000]	
	:5326		
U 1C2, 0000.00	:5327	WORD[0000]	:LOCATION 0AF
U 1C3, 0000.00	:5328	WORD[0000]	
	:5329		
U 1C4, 0000.00	:5330	WORD[0000]	:LOCATION 0B0
U 1C5, 0000.00	:5331	WORD[0000]	

Address	Value	Label
U 1C6, 0000,00	WORD[0000]	:LOCATION 0B1
U 1C7, 0000,00	WORD[0000]	
U 1C8, 0000,00	WORD[0000]	:LOCATION 0B2
U 1C9, 0000,00	WORD[0000]	
U 1CA, 0000,00	WORD[0000]	:LOCATION 0B3
U 1CB, 0000,00	WORD[0000]	
U 1CC, 0000,00	WORD[0000]	:LOCATION 0B4
U 1CD, 0000,00	WORD[0000]	
U 1CE, 0000,00	WORD[0000]	:LOCATION 0B5
U 1CF, 0000,00	WORD[0000]	
U 1D0, 0000,00	WORD[0000]	:LOCATION 0B6
U 1D1, 0000,00	WORD[0000]	
U 1D2, 0000,00	WORD[0000]	:LOCATION 0B7
U 1D3, 0000,00	WORD[0000]	
U 1D4, 0000,00	WORD[0000]	:LOCATION 0B8
U 1D5, 0000,00	WORD[0000]	
U 1D6, 00FF,FF	WORD[0FFFF]	:LOCATION 0B9
U 1D7, 0000,00	WORD[0000]	
U 1D8, 0000,86	WORD[0086]	:LOCATION 0BA
U 1D9, 0000,00	WORD[0000]	
U 1DA, 0000,00	WORD[0000]	:LOCATION 0BB
U 1DB, 00FF,F8	WORD[0FFF8]	
U 1DC, 0002,C4	WORD[02C4]	:LOCATION 0BC
U 1DD, 0028,80	WORD[2880]	
U 1DE, 0001,8A	WORD[018A]	:LOCATION 0BD
U 1DF, 0012,40	WORD[1240]	
U 1E0, 0003,86	WORD[0386]	:LOCATION 0BE
U 1E1, 000A,00	WORD[0A00]	
U 1E2, 0000,CE	WORD[00CE]	:LOCATION 0BF
U 1E3, 003A,00	WORD[3A00]	
U 1E4, 0003,CE	WORD[03CE]	:LOCATION 0C0
U 1E5, 0000,40	WORD[0040]	
U 1E6, 00FF,7F	WORD[0FFF7F]	:LOCATION 0C1
U 1E7, 00FF,FF	WORD[0FFFF]	
U 1E8, 00FF,FF	WORD[0FFFF]	:LOCATION 0C2
U 1E9, 00FF,BF	WORD[0FFBF]	

D 10
19-MAY-83 13:28:52
Fiche 1 Frame D10
Sequence 120
Page 117

ZZ-ENKCH-1.0 : ENKCH.MIC MICRO2 1M(01) PROGRAM SPECIFIC DATA SECTION (LS 80-F7)
: ENKCH.MCR
: ENKCH.MIC

U 1EA, 00FF,FF	:5387	WORD[0FFFF]	:LOCATION 0C3
U 1EB, 00FF,7F	:5388	WORD[0FF7F]	
	:5389		
U 1EC, 0000,03	:5390	WORD[0003]	:LOCATION 0C4
U 1ED, 0000,00	:5391	WORD[0000]	
	:5392		
U 1EE, 0000,FE	:5393	WORD[00FE]	:LOCATION 0C5
U 1EF, 0000,00	:5394	WORD[0000]	
	:5395		
U 1F0, 0001,FE	:5396	WORD[01FE]	:LOCATION 0C6
U 1F1, 0000,00	:5397	WORD[0000]	
	:5398		
U 1F2, 0000,00	:5399	WORD[0000]	:LOCATION 0C7
U 1F3, 0000,00	:5400	WORD[0000]	
	:5401		
U 1F4, 00FF,F0	:5402	WORD[0FFF0]	:LOCATION 0C8
U 1F5, 00FF,FF	:5403	WORD[0FFFF]	
	:5404		
U 1F6, 0000,06	:5405	WORD[0006]	:LOCATION 0C9
U 1F7, 0000,00	:5406	WORD[0000]	
	:5407		
U 1F8, 00FF,FF	:5408	WORD[0FFFF]	:LOCATION 0CA
U 1F9, 0000,07	:5409	WORD[0007]	
	:5410		
U 1FA, 0000,0F	:5411	WORD[000F]	:LOCATION 0CB
U 1FB, 0000,00	:5412	WORD[0000]	
	:5413		
U 1FC, 00FC,FF	:5414	WORD[0FCFF]	:LOCATION 0CC
U 1FD, 00FF,FF	:5415	WORD[0FFFF]	
	:5416		
U 1FE, 00FF,FC	:5417	WORD[0FFFC]	:LOCATION 0CD
U 1FF, 00FF,FF	:5418	WORD[0FFFF]	
	:5419		
U 200, 00FF,80	:5420	WORD[0FF80]	:LOCATION 0CE
U 201, 00FF,FF	:5421	WORD[0FFFF]	
	:5422		
U 202, 0000,00	:5423	WORD[0000]	:LOCATION 0CF
U 203, 0020,00	:5424	WORD[2000]	
	:5425		
U 204, 0005,00	:5426	WORD[0500]	:LOCATION 0D0
U 205, 0020,10	:5427	WORD[2010]	
	:5428		
U 206, 0005,00	:5429	WORD[0500]	:LOCATION 0D1
U 207, 00A0,10	:5430	WORD[0A010]	
	:5431		
U 208, 00FF,FF	:5432	WORD[0FFFF]	:LOCATION 0D2
U 209, 00FF,CF	:5433	WORD[0FFCF]	
	:5434		
U 20A, 00E0,00	:5435	WORD[0E000]	:LOCATION 0D3
U 20B, 00FF,FF	:5436	WORD[0FFFF]	
	:5437		
U 20C, 0000,69	:5438	WORD[0069]	:LOCATION 0D4
U 20D, 0000,00	:5439	WORD[0000]	
	:5440		
U 20E, 0068,37	:5441	WORD[6837]	:LOCATION 0D5

U 20F, 0000,00	:5442	WORD[0000]	
	:5443		
U 210, 0072,00	:5444	WORD[7200]	:LOCATION 0D6
U 211, 0000,00	:5445	WORD[0000]	
	:5446		
U 212, 00F8,00	:5447	WORD[0F800]	:LOCATION 0D7
U 213, 00FF,FF	:5448	WORD[0FFFF]	
	:5449		
U 214, 0010,21	:5450	WORD[1021]	:LOCATION 0D8
U 215, 0000,00	:5451	WORD[0000]	
	:5452		
U 216, 0005,00	:5453	WORD[0500]	:LOCATION 0D9
U 217, 00A0,01	:5454	WORD[0A001]	
	:5455		
U 218, 0005,00	:5456	WORD[0500]	:LOCATION 0DA
U 219, 0020,01	:5457	WORD[2001]	
	:5458		
U 21A, 0000,1F	:5459	WORD[001F]	:LOCATION 0DB
U 21B, 0000,00	:5460	WORD[0000]	
	:5461		
U 21C, 0001,69	:5462	WORD[0169]	:LOCATION 0DC
U 21D, 0000,00	:5463	WORD[0000]	
	:5464		
U 21E, 0001,01	:5465	WORD[0101]	:LOCATION 0DD
U 21F, 0000,00	:5466	WORD[0000]	
	:5467		
U 220, 00FF,FF	:5468	WORD[0FFFF]	:LOCATION 0DE
U 221, 00FF,F0	:5469	WORD[0FFF0]	
	:5470		
U 222, 0000,00	:5471	WORD[0000]	:LOCATION 0DF
U 223, 0000,03	:5472	WORD[0003]	
	:5473		
U 224, 0000,00	:5474	WORD[0000]	:LOCATION 0E0
U 225, 0000,07	:5475	WORD[0007]	
	:5476		
U 226, 0000,00	:5477	WORD[0000]	:LOCATION 0E1
U 227, 0000,0F	:5478	WORD[000F]	
	:5479		
U 228, 00EB,FF	:5480	WORD[0EBFF]	:LOCATION 0E2
U 229, 00FF,FF	:5481	WORD[0FFFF]	
	:5482		
U 22A, 006B,FF	:5483	WORD[6BFF]	:LOCATION 0E3
U 22B, 00FF,FF	:5484	WORD[0FFFF]	
	:5485		
U 22C, 0094,00	:5486	WORD[9400]	:LOCATION 0E4
U 22D, 0000,00	:5487	WORD[0000]	
	:5488		
U 22E, 006F,FF	:5489	WORD[6FFF]	:LOCATION 0E5
U 22F, 00FF,FF	:5490	WORD[0FFFF]	
	:5491		
U 230, 0090,00	:5492	WORD[9000]	:LOCATION 0E6
U 231, 0000,00	:5493	WORD[0000]	
	:5494		
U 232, 00EF,FF	:5495	WORD[0EFFF]	:LOCATION 0E7
U 233, 00FF,FF	:5496	WORD[0FFFF]	

U
U
U
U


```

:5547 .PAGE 'WCS SUBROUTINES FOR CONSOLE USE'
:5548 .TOC 'GENERATE TRANSFER DATA'
:5549 :
:5550 This routine is used by the diagnostic monitor to load WR 0 with a
:5551 data pattern from the console. The monitor will set CONSOLE ATTN if
:5552 a 1 is to be generated, or clear CONSOLE ATTN if a 0 is to be gen-
:5553 erated. It will then single step this routine to shift the data bit
:5554 into WR 0. This routine loads a 32 bit value much more quickly than
:5555 shifting each instruction into the CSR from the monitor, and is used
:5556 mainly to set up data to load in LS for constants.
:5557 :
:5558 :
:5559 ***WARNING***
:5560 :
:5561 This routine must start at 600 and end at 605. DO NOT move this
:5562 routine to any other area!!
:5563 :
:5564 :
:5565 :
:5566 GEN.XFER.DATA:
:5567 CLR WR[0] ;CLEAR A WORKING REG
:5568 COM WR[0] ;NOW WR 0 IS ALL ONES
:5569 :
:5570 LOOP.XFER:
:5571 SKIP.IF[CONSOLE.ATTN] ;SKIP NEXT INSTRUCTION IS CONSOLE ATTN
:5572 : ;IS SET (GENERATING A 1)
:5573 ASHL WR[0], ;SHIFT A 0 INTO BIT 0 OF WR 0
:5574 SKIP ;AND SKIP ROL
:5575 ROL WR[0] ;SHIFT A 1 INTO BIT 0 OF WR 0
:5576 JMP [LOOP.XFER] ;REPEAT
:5577 :
:5578 :
:5579 DIAGNOSTIC VERSION NUMBER IS STORED HERE
:5580 :
:5581 :
:5582 ***WARNING***
:5583 :
:5584 These words must be at location 606 and 607. DO NOT move this word
:5585 to any other area!!
:5586 :
:5587 :
:5588 :
:5589 WORD[0100] ;VERSION 01.00
:5590 ASCII [CH] ;LAST 2 LETTERS OF FILE NAME [ENKCH]
:5591 :
:5592 :
:5593 :
    
```

U 600, 2F80,15
 U 601, A0C0,15
 U 602, 5B00,18
 U 603, A3D0,1E
 U 604, A3C0,15
 U 605, 0860,24
 U 606, 0001,00
 U 607, 0043,48

```

:5594 .PAGE  " DEPOSIT MCT CSR ROUTINE"
:5595 .+++
:5596
:5597 .FUNCTIONAL DESCRIPTION:
:5598
:5599 .The deposit to CSR routine is used to write the memory controller's
:5600 .control and status register. The memory controller has three CSRs
:5601 .named CSR 0, CSR 1, and CSR 2.
:5602 .CSR 0 contains checkbits or syndrome bits returned from the ECC logic
:5603 .after certain diagnostic routines. It is NOT writable directly with
:5604 .this routine.
:5605 .CSR 1 contains error bits set if an error occurs on a memory access.
:5606 .It does contain five maintenance bits (bits 29-25) which are writable.
:5607 .There are also seven checkbits (bits 6-0) which are writable, but not
:5608 .readable. These bits are used to write checkbits into the ECC chips.
:5609 .CSR 2 contains error bits set if a UNIBUS error occurs on a UNIBUS
:5610 .access. These are read only bits and are NOT writable by this routine.
:5611 .Therefore, CSR 1 is the only writable CSR, and only bits 29-25 can be
:5612 .both written and read back. This routine will thus force CSR 1 on a
:5613 .deposit to CSR.
:5614 .This routine will write LS 5 with data to access CSR 1, then write
:5615 .the data residing in LS 6 into the CSR. It will then issue CPU ATTN
:5616 .to inform the console that it has completed the function.
:5617 .NOTE: The error bits of CSR 1 are cleared on any write to CSR 1.
:5618 .These are bits 31, 30 and 23-14. Bits 13-0 and bit 24 are unused.
:5619
:5620 .CALLING SEQUENCE:
:5621
:5622 .The console loads the address of a pointer to this routine (a JMP in-
:5623 .struction at WCS location 50) into the UPC and starts the CPU.
:5624
:5625 .INPUT PARAMETERS:
:5626
:5627 .LS location 6 must have been written by the console with the desired
:5628 .data to be deposited in CSR 1 previous to executing this routine.
:5629
:5630 .IMPLICIT INPUTS:
:5631
:5632 .None
:5633
:5634 .OUTPUT PARAMETERS:
:5635
:5636 .CSR 1 bits 29-25 and 6-0 are written with data from LS 6.
:5637
:5638 .IMPLICIT OUTPUTS:
:5639
:5640 .None
:5641
:5642 .SIDE EFFECTS:
:5643
:5644 .WR 0 will be modified by this routine.
:5645 .LS 5 will be modified by this routine.
:5646 .CSR 1 bits 31, 30 and 23-14 are cleared.
:5647
:5648 .---
```

```

:5649
:5650 DEPOSIT.CSR:
:5651
U 608, 3644,15 :5652 MOV LS[CSR1] TO WR[0] ;SET BIT 2 IN WR 0 AS CSR NUMBER TO
:5653 ; WRITE (BITS 2 AND 3 ARE CSR NUMBER)
U 609, BE0A,15 :5654 MOV WR[0] TO LS[T5.SUB] ;PUT CSR NUMBER IN LS LOCATION 5 (CSR
:5655 ; 1)
U 60A, 1DOB,F5 :5656 MEM.REQ[WRITE.CSR] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:5657 ; ACCESS CSR 1
U 60B, B20C,15 :5658 WRITE.MEM LS[T6.SUB] ;SEND DATA IN LS LOCATION 6 TO CSR 1
U 60C, 8868,74 :5659 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT

```



```

:5660 .PAGE  " EXAMINE MCT CSR ROUTINE"
:5661 .+++
:5662
:5663 .FUNCTIONAL DESCRIPTION:
:5664
:5665     The examine CSR routine reads the memory controller's control and
:5666     status register. There are 3 CSRs labeled CSR 0, CSR 1 and CSR 2.
:5667     CSR 0 contains checkbits or syndrome bits from the ECC logic.
:5668     These are contained in bits 6-0 of the CSR. Other bits are UNPRE-
:5669     DICTABLE. Bits 6-0 are only written by special diagnostic routines
:5670     (they are not writable by the DEPOSIT CSR command).
:5671     CSR 1 contains error bits and maintenance bits. The error bits are
:5672     31, 30, and 23-14. Maintenance bits are 29-25. The other bits are
:5673     UNPREDICTABLE. The error bits are written during memory functions
:5674     only and are not writable via the DEPOSIT CSR command. The maintenance
:5675     bits are writable. Note that a DEPOSIT CSR command will write the
:5676     maintenance bits and clear all error bits.
:5677     CSR 2 contains three bits (bits 16-14) which are readable. These are
:5678     set when a UNIBUS error occurs on a UNIBUS access. They are not
:5679     directly writable with the DEPOSIT CSR command.
:5680     The console loads LS 5 with the CSR number (0, 1, or 2) to be read.
:5681     This routine rotates LS 5 two places left to position it correctly.
:5682     It then executes a read CSR and places the data in LS 6 for the console
:5683     to read.
:5684
:5685 .CALLING SEQUENCE:
:5686
:5687     The console loads the address of a pointer to this routine (a JMP in-
:5688     struction at WCS location 51) into the UPC and starts the CPU.
:5689
:5690 .INPUT PARAMETERS:
:5691
:5692     LS 5 contains the CSR number to be read.
:5693
:5694 .IMPLICIT INPUTS:
:5695
:5696     None
:5697
:5698 .OUTPUT PARAMETERS:
:5699
:5700     LS 6 - Data read from CSR selected.
:5701
:5702 .IMPLICIT OUTPUTS:
:5703
:5704     None
:5705
:5706 .SIDE EFFECTS:
:5707
:5708     LS 5 is rotated 2 places left.
:5709
:5710 .---
:5711
:5712 .EXAMINE.CSR:
:5713
:5714     MOV LS[T5.SUB] TO WR[0]           ;GET CSR NUMBER FROM LS 5
    
```

K 10
Fiche 1 Frame K10
Sequence 127
Page 124

ZZ-ENKCH-1.0 : ENKCH.MIC EXAMINE MCT CSR ROUTINE
 : ENKCH.MCR MICRO2 1M(01) 19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module Rev 01.00
 : ENKCH.MIC EXAMINE MCT CSR ROUTINE

```

U 60E, A2D0,15 :5715      SHL2 WR[0]                ;SHIFT NUMBER 2 PLACES LEFT TO GET
                  :5716                ; CSR NUMBER IN BITS 2 AND 3
U 60F, BE0A,15 :5717      MOV WR[0] TO LS[T5.SUB]      ;PUT SHIFTED NUMBER IN LS 5
                  :5718
U 610, 9D0B,75 :5719      MEM.REQ[READ.CSR] ADRS[T5.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
                  :5720                ; ACCESS CSR SPECIFIED
U 611, 3022,15 :5721      MOV MEM.DATA TO WR[0]          ;READ CSR SELECTED
                  :5722
                  :5723
U 612, B60A,95 :5724      CHECK.CSR2:
                  :5725      MOV LS[T5.SUB] TO WR[1]      ;GET SHIFTED CSR NUMBER
                  :5726      CMP LS[#8] WITH WR[1],        ;SEE IF CSR 2 WAS SELECTED
                  :5727      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
                  :5728      JMP.IF[NEQ] TO [CHECK.CSR1]    ;JUMP IF CSR 2 WAS NOT SELECTED
U 613, CE46,B5 :5729      BIC LS[CSR2.MASK] TO WR[0]      ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  :5730                ; CSR 2.
                  :5731
U 616, 4E44,B5 :5732      CHECK.CSR1:
                  :5733      CMP LS[#4] WITH WR[1],        ;SEE IF CSR 1 WAS SELECTED
                  :5734      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
U 617, 0861,91 :5735      JMP.IF[NEQ] TO [CHECK.CSR0]      ;JUMP IF CSR 1 WAS NOT SELECTED
U 618, C52E,15 :5736      BIC LS[CSR1.MASK] TO WR[0]      ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  :5737                ; CSR 1.
                  :5738
U 619, 4E9C,B5 :5739      CHECK.CSR0:
                  :5740      CMP LS[#0] WITH WR[1],        ;SEE IF CSR 0 WAS SELECTED
                  :5741      DT(LONG)&SET.ALU.CC          ;SET UP CONDITION CODES
U 61A, 8861,D1 :5742      JMP.IF[NEQ] TO [LOAD.LS6]        ;JUMP IF CSR 0 WAS NOT SELECTED
U 61B, 452C,15 :5743      BIC LS[CSR0.MASK] TO WR[0]      ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  :5744                ; CSR 0.
                  :5745      BIC LS[BIT7] TO WR[0]        ;BIT 7 IS ALSO UNUSED
U 61C, C54E,15 :5746
                  :5747
U 61D, BE0C,15 :5748      LOAD.LS6:
U 61E, 8868,74 :5749      MOV WR[0] TO LS[T6.SUB]      ;LOAD CSR DATA INTO LS 6 FOR PRINTOUT
                  :5750      JMP [WAIT.SUB]              ;ISSUE CPU ATTN AND WAIT FOR RESPONSE
  
```

:5747 :page " DEPOSIT MCT TRANSLATION BUFFER ROUTINE"
 :5748 :+++

:5750 :FUNCTIONAL DESCRIPTION:

:5751 :
 :5752 :This routine allows a write to the translation buffer portion of the
 :5753 :memory controller. The buffer area contains 128 decimal locations
 :5754 :addressed from 0-7F(H). The remainder of the TB RAM is either unused
 :5755 :or contains the UNIBUS MAP. The DEPOSIT UB command is used to write
 :5756 :the UNIBUS map portion of the TB RAMs. The address specified with the
 :5757 :DEPOSIT command will be the actual RAM address accessed. This routine
 :5758 :will do any shifting necessary on the address specified to position it
 :5759 :correctly. The address specified has been loaded by the console in LS
 :5760 :location 5 prior to executing this routine. The address must be in HEX
 :5761 :format.

:5762 :The data specified has been placed in LS 6 by the console prior to
 :5763 :executing this routine. Bits 07-01 will be the Valid, Prot, Modify,
 :5764 :and Byte Offset bits (there are four Prot bits). Bits 22-08 will
 :5765 :contain the PA bits from PA 23 through PA 09 respectively. Bits 31
 :5766 :through 23 and bit 0 are unused. This routine will do any shifting nec-
 :5767 :essary to write the bits in TB as described. The data specified must
 :5768 :be in HEX format.

:5769 :
:5770 :CALLING SEQUENCE:

:5771 :
 :5772 :The console loads the address of a pointer to this routine (a JMP in-
 :5773 :struction at WCS location 52) into the UPC and starts the CPU.

:5774 :
:5775 :INPUT PARAMETERS:

:5776 :
 :5777 :LS 5 - TB address to be written (range 0-7F)
 :5778 :LS 6 - TB data to write (22 bits, from 1-22). Bits 0 and 23-31 are
 :5779 :ignored.

:5780 :
:5781 :IMPLICIT INPUTS:

:5782 :None

:5783 :
:5784 :OUTPUT PARAMETERS:

:5785 :TB will be written with specified data at specified address

:5786 :
:5787 :IMPLICIT OUTPUTS:

:5788 :None

:5789 :
:5790 :SIDE EFFECTS:

:5791 :
 :5792 :WR 0 will be modified
 :5793 :WR 1 will be modified

:5794 :
:5795 :---

U 61F, 8862,FC :5800 DEPOSIT.TB:
 :5801 JSR [SHIFT.TB.ADR] ;GO TO SUBROUTINE TO POSITION TB AD-

ZZ-ENKCH-1.0	:	ENKCH.MIC	M 10	
:	:	MICRO2 1M(01)	19-MAY-83	Fiche 1 Frame M10
:	:	DEPOSIT MCT TRANSLATION BUFFER ROUTINE	13:28:52	Sequence 129
			ENKCH Micro-diagnostic for IDC module	Page 126
			Rev 01.00	

U 620, 360C,15	:5802		: DRESS CORRECTLY
U 621, 4526,15	:5803	MOV LS[T6.SUB] TO WR[0]	: GET DATA FROM LS 6
	:5804	BIC LS[FFFF] TO WR[0]	: CLEAR LOW BYTE OF DATA SPECIFIED.
	:5805		: LOW BYTE WILL BE PUT IN BITS 24-31
	:5806		: LATER
U 622, 8862,AC	:5807	JSR [SHIFT.LOOP]	: GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
	:5808		: PLACES RIGHT
U 623, E40C,15	:5809	SWAP LS[T6.SUB] WITH WR[0]	: SWAP ORIGINAL DATA SPECIFIED WITH MOD-
	:5810		: IFIED DATA. NOW LS 6 CONTAINS BITS
	:5811		: 8-22 IN BITS 0-14 AS REQUIRED. WR 0
	:5812		: CONTAINS ORIGINAL DATA SPECIFIED
U 624, 452C,15	:5813	BIC LS[FFFFFF00] TO WR[0]	: CLEAR ALL BUT LOW BYTE OF ORIGINAL
	:5814		: DATA SPECIFIED
U 625, 8862,AC	:5815	JSR [SHIFT.LOOP]	: GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
	:5816		: PLACES RIGHT. ON RETURN, WR 0 WILL
	:5817		: HAVE BITS 0-7 IN BITS 24-31, OTHER
	:5818		: BITS ARE CLEAR
U 626, EDOC,15	:5819	BIS WR[0] TO LS[T6.SUB]	: NOW LS 6 CONTAINS DATA TO WRITE IN
	:5820		: TB IN CORRECT FORMAT I.E. PA BITS
	:5821		: IN BITS 00-14 AND BYTE OFFSET, MODIFY
	:5822		: PROT A THROUGH PROT D, AND VALID IN
	:5823		: BITS 25-31.
	:5824		
U 627, 980A,F5	:5825	MEM.REQ[WRITE.TB] ADRS[T5.SUB] DT[LONG]	: ISSUE MEMORY REQUEST TO WRITE
	:5826		: THE TB
U 628, B20C,15	:5827	WRITE.MEM LS[T6.SUB]	: WRITE DATA FROM LS 6 IN TB
U 629, 8868,74	:5828	JMP [WAIT.SUB]	: JUMP TO SET ATTN AND WAIT
	:5829		
	:5830		
	:5831		
	:5832		
	:5833		
U 62A, 3646,95	:5834	SHIFT.LOOP:	
	:5835	MOV LS[#8] TO WR[1]	: COUNTER IN WR 1 TO SHIFT 8 PLACES
U 62B, 2340,15	:5836	SHIFT.LOOP.1:	
	:5837	ROR WR[0]	: SHIFT WR 0 ONE PLACE RIGHT
	:5838	DEC WR[1]	: DECREMENT COUNTER
U 62C, A102,B5	:5839	DT[LONG]&SET.ALU.CC	: AND SET CONDITION CODES
U 62D, 8862,B1	:5840	JMP.IF[NEQ] TO [SHIFT.LOOP.1]	: REPEAT UNTIL 8 SHIFTS HAVE OCCURRED
U 62E, 5B00,14	:5841	RETURN	: THEN RETURN
	:5842		
	:5843		
	:5844		
	:5845		
	:5846		
U 62F, 360A,15	:5847	SHIFT.TB.ADR:	
U 630, 452C,15	:5848	MOV LS[T5.SUB] TO WR[0]	: GET TB ADDRESS FROM LS 5
U 631, A2D0,15	:5849	BIC LS[FFFFFF00] TO WR[0]	: CLEAR ALL BUT LOW BYTE (8 BIT ADDRESS)
U 632, A2D0,15	:5850	SHL2 WR[0]	: SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 633, A2D0,15	:5851	SHL2 WR[0]	: SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 634, A2D0,15	:5852	SHL2 WR[0]	: SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 635, 23D0,15	:5853	ASHL WR[0]	: SHIFT ONE MORE PLACE
	:5854		: NOW BITS 0-6 IN BITS 9-15
	:5855		: SEE IF BIT 15 (MSB) IS SET OR CLEAR.
U 636, 595E,35	:5856	BIT LS[BIT15] WITH WR[0],	: MSB MUST GO IN BIT 31 IN ORDER TO AD-
		DT[LONG]&SET.ALU.CC	: DRESS TB CORRECTLY

ZZ-ENKCH-1.0	:	ENKCH.MIC		N 10				
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	Fiche 1	Frame N10	Sequence 130
:	:	ENKCH.MIC	DEPOSIT MCT TRANSLATION BUFFER ROUTINE	ENKCH	Micro-diagnostic for IDC module	Rev 01.00		Page 127

U 637, 5B00,09	:	5857	SKIP.IF[EQL]	:	SKIP NEXT INSTRUCTION IF MSB IS 0
U 638, 477E,15	:	5858	BIS LS[BIT31] TO WR[0]	:	ELSE SET BIT 31 FOR MSB
	:	5859			
	:	5860	MOV WR[0] TO LS[T5.SUB],	:	NOW TB ADDRESS IS IN CORRECT FORM IN
	:	5861		:	LS LOCATION 5
U 639, 3E0A,14	:	5862	RETURN	:	THEN RETURN TO MAIN CODE

```

5863 PAGE " EXAMINE TRANSLATION BUFFER ROUTINE"
5864 ***
5865
5866 FUNCTIONAL DESCRIPTION:
5867
5868 This routine will read the translation buffer of the memory controller.
5869 The TB will be returned with the byte offset, modify, prot A through
5870 prot D, and valid in bits 01-07 respectively. PA bits 09-23 are re-
5871 turned in bits 08-22 respectively. Bits 23-31 and bit 0 are not used
5872 and so will be cleared before printing the result. The result will be
5873 put in LS 6 for the console to print.
5874 The routine will issue a memory request with memory function=READ.TB
5875 and data type=LONGWORD. The console will load LS 5 with the address
5876 specified before executing this routine. The routine will shift the
5877 address around as required to address the TB location specified. The
5878 address specified must be in HEX format.
5879 The TB consists of 128 decimal locations (addresses from 0-7F). The
5880 remaining locations in the TB RAM are either unused or used by the
5881 UNIBUS map. UNIBUS map addresses are read via the EXAMINE UB routine.
5882
5883 CALLING SEQUENCE:
5884
5885 The console loads the address of a pointer to this routine (a JMP in-
5886 struction at WCS location 53) into the UPC and starts the CPU.
5887
5888 INPUT PARAMETERS:
5889
5890 LS 5 - TB address (0-7F) in hex format
5891
5892 IMPLICIT INPUTS:
5893
5894 None
5895
5896 OUTPUT PARAMETERS:
5897
5898 LS 6 - TB data from address specified
5899
5900 IMPLICIT OUTPUTS:
5901
5902 None
5903
5904 SIDE EFFECTS:
5905
5906 LS 5 is modified
5907 WR 0 is modified
5908
5909 ---
5910
5911 EXAMINE.TB:
5912 JSR [SHIFT.TB.ADR] ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
5913 ; CORRECT POSITION AND REPLACE IN LS 5
5914 MEM.REQ[READ.TB] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO READ
5915 ; THE TB AT ADDRESS CONTAINED IN LS 5
5916 MOV MEM.DATA TO WR[0] ;PLACE RESULT IN WR 0
5917

```

U 63A, 8862,FC

U 63B, 9C0B, F5

U 63C, 3022,15

www.pearsoned.com

5926 PAGE " DEPOSIT UNIBUS MAP ROUTINE"

5927 +++

5928 FUNCTIONAL DESCRIPTION:

5929 This routine allows a write to the UNIBUS map portion of the TB on the
 5930 memory controller. The map area contains 512 decimal locations
 5931 addressed from 200-3FF. The remainder of the TB RAM is either unused
 5932 or contains the Translation Buffer information. The DEPOSIT TB command
 5933 is used to write the TB portion of the TB RAMs. The address specified
 5934 with the DEPOSIT command will be the actual RAM address accessed. This
 5935 routine will do any shifting necessary on the address specified to pos-
 5936 sition it correctly. The address specified has been loaded by the con-
 5937 sole in LS location 5 prior to executing this routine. The address
 5938 must be in HEX format.

5939 The data specified has been placed in LS 6 by the console prior to
 5940 executing this routine. Bits 07-01 will be the Valid, Prot, Modify,
 5941 and Byte Offset bits (there are four Prot bits). Bits 22-08 will
 5942 contain the PA bits from PA 23 through PA 09 respectively. Bits 31
 5943 through 23 and bit 0 are unused. This routine will do any shifting nec-
 5944 essary to write the bits in TB as described. The data specified must
 5945 be in HEX format.

5946 CALLING SEQUENCE:

5947 The console loads the address of a pointer to this routine (a JMP in-
 5948 struction at WCS location 58) into the UPC and starts the CPU.

5949 INPUT PARAMETERS:

5950 LS 5 - UNIBUS Map address in TB (must be in range 200-3FF HEX).
 5951 LS 6 - Contains the data to write into the UNIBUS Map in bits 1-22.

5952 IMPLICIT INPUTS:

5953 None

5954 OUTPUT PARAMATERS:

5955 The UNIBUS map portion of the TB RAMs gets written with the data from
 5956 LS 6 at the address specified in LS 5.

5957 IMPLICIT OUTPUTS:

5958 None

5959 SIDE EFFECTS:

5960 WR 0 will be modified
 5961 WR 1 will be modified

5962 ---

5963 DEPOSIT.UBS:

5964 JSR [SHIFT.UB.ADR]

;GO TO SUBROUTINE TO ARRANGE UNIBUS

U 642, 0864,DC

5965

U 643, 360C,15	:5981		: MAP ADDRESS CORRECTLY
U 644, 4526,15	:5982	MOV LS[T6.SUB] TO WR[0]	: GET DATA FROM LS 6
	:5983	BIC LS[##FF] TO WR[0]	: CLEAR LOW BYTE OF DATA SPECIFIED.
	:5984		: LOW BYTE WILL BE PUT IN BITS 24-31
	:5985		: LATER
U 645, 8862,AC	:5986	JSR [SHIFT.LOOP]	: GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
	:5987		: PLACES RIGHT
U 646, E40C,15	:5988	SWAP LS[T6.SUB] WITH WR[0]	: SWAP ORIGINAL DATA SPECIFIED WITH MOD-
	:5989		: IFIED DATA. NOW LS 6 CONTAINS BITS
	:5990		: 8-22 IN BITS 0-14 AS REQUIRED. WR 0
	:5991		: CONTAINS ORIGINAL DATA SPECIFIED
U 647, 452C,15	:5992	BIC LS[#####F00] TO WR[0]	: CLEAR ALL BUT LOW BYTE OF ORIGINAL
	:5993		: DATA SPECIFIED
U 648, 8862,AC	:5994	JSR [SHIFT.LOOP]	: GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
	:5995		: PLACES RIGHT. ON RETURN, WR 0 WILL
	:5996		: HAVE BITS 0-7 IN BITS 24-31, OTHER
	:5997		: BITS ARE CLEAR
U 649, ED0C,15	:5998	BIS WR[0] TO LS[T6.SUB]	: NOW LS 6 CONTAINS DATA TO WRITE IN
	:5999		: TB IN CORRECT FORMAT I.E. PA BITS
	:6000		: IN BITS 00-14 AND BYTE OFFSET, MODIFY
	:6001		: PROT A THROUGH PROT D, AND VALID IN
	:6002		: BITS 25-31.
	:6003		
U 64A, 1B0B,F5	:6004	MEM.REQ[WRITE.UBS.MAP] ADRS[T5.SUB] DT[LONG]	: ISSUE MEMORY REQUEST TO
	:6005		: WRITE THE UNIBUS MAP
U 64B, B20C,15	:6006	WRITE.MEM LS[T6.SUB]	: WRITE DATA FROM LS 6 IN UNIBUS MAP
U 64C, 8868,74	:6007	JMP [WAIT.SUB]	: JUMP TO SET ATTN AND WAIT
	:6008		
	:6009		
	:6010	SUBROUTINE TO POSITION ADDRESS SPECIFIED CORRECTLY TO ADDRESS THE	
	:6011	UNIBUS MAP.	
	:6012		
	:6013	SHIFT.UB.ADR:	
U 64D, 360A,15	:6014	MOV LS[T5.SUB] TO WR[0]	: GET ADDRESS SPECIFIED IN WR 0
U 64E, A2D0,15	:6015	SHL2 WR[0]	: SHIFT ADDRESS 2 PLACES LEFT
U 64F, A2D0,15	:6016	SHL2 WR[0]	: SHIFT ADDRESS 2 PLACES LEFT
U 650, A2D0,15	:6017	SHL2 WR[0]	: SHIFT ADDRESS 2 PLACES LEFT
U 651, A2D0,15	:6018	SHL2 WR[0]	: SHIFT ADDRESS 2 PLACES LEFT
U 652, 23D0,15	:6019	ASHL WR[0]	: SHIFT LEFT ONE MORE PLACE
	:6020		: NOW ADDRESS IS IN BITS 9-17 OF WR 0
	:6021	MOV WR[0] TO LS[T5.SUB],	: NOW LS 5 CONTAINS CORRECT ADDRESS.
U 653, 3E0A,14	:6022	RETURN	: RETURN TO MAIN
	:6023		


```

:6024 .PAGE " EXAMINE UNIBUS MAP ROUTINE"
:6025 .+++
:6026 This routine will read the UNIBUS map portion of the memory controller.
:6027 The UNIBUS MAP contents will be returned with the byte offset, modify,
:6028 prot A through prot D, and valid in bits 01-07 respectively. PA bits
:6029 09-23 are returned in bits 08-22 respectively. Bits 23-31 and bit 0
:6030 are not used and so will be cleared before printing the result. The
:6031 result will be put in LS 6 for the console to print.
:6032 The routine will issue a memory request with memory function=READ.UBS
:6033 .MAP and data type=LONGWORD. The console will load LS 5 with the ad-
:6034 dress specified before executing this routine. The routine will shift
:6035 the address around as required to address the UNIBUS map location spec-
:6036 ified. The address specified must be in HEX format.
:6037 The UNIBUS map consists of 512 decimal locations (addresses from 200-
:6038 3FF) The remaining locations in the TB RAM are either unused or used
:6039 by the translation buffer. TB contents are read via the EXAMINE TB
:6040 routine.
:6041
:6042 CALLING SEQUENCE:
:6043
:6044 The console loads the address of a pointer to this routine (a JMP in-
:6045 struction at WCS location 59) into the UPC and starts the CPU.
:6046
:6047 INPUT PARAMETERS:
:6048
:6049 LS 5 - TB address in hex format
:6050
:6051 IMPLICIT INPUTS:
:6052
:6053 None
:6054
:6055 OUTPUT PARAMETERS:
:6056
:6057 LS 6 - UNIBUS map data from address specified
:6058
:6059 IMPLICIT OUTPUTS:
:6060
:6061 None
:6062
:6063 SIDE EFFECTS:
:6064
:6065 LS 5 is modified
:6066 WR 0 will be modified
:6067
:6068 ---
:6069
:6070 EXAMINE.UBS:
:6071 JSR [SHIFT.UB.ADR] ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
:6072 ; CORRECT POSITION AND REPLACE IN LS 5
:6073 MEM.REQ[READ.UBS.MAP] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:6074 ; READ THE UBS MAP AT ADDRESS CONTAINED
:6075 ; IN LS 5
:6076 MOV MEM.DATA TO WR[0] ;PLACE RESULT IN WR 0
:6077
:6078 BIC LS[#FF000000] TO WR[0] ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE

```

U 654, 0864,DC

U 655, 1C0B,75

U 656, 3022,15

U 657, 452A,15

U 658, 456E,15	:6079		: IS NOT PART OF TB AND IS UNPREDICT-
U 659, 4540,15	:6080		: ABLE)
	:6081	BIC LS[BIT23] TO WR[0]	:DITTO FOR BIT 23
	:6082	BIC LS[BIT0] TO WR[0]	:DITTO FOR BIT 0. WR 0 NOW CONTAINS
	:6083		: RESULT TO PRINT
U 65A, BE0C,15	:6084	MOV WR[0] TO LS[6.SUB]	:RESULT NOW IN LS 6
U 65B, 8868,74	:6085	JMP [WAIT.SUB]	:JUMP TO SET ATTN AND WAIT
	:6086		: CONTROL

```

:6087 .PAGE  " DEPOSIT MAIN MEMORY ROUTINE"
:6088 .+++
:6089
:6090 .FUNCTIONAL DESCRIPTION:
:6091
:6092     This routine write a longword of data into main memory at the Physical
:6093     address specified. This address need not be on a longword boundary.
:6094     The console will use this routine for data transfers to main memory as
:6095     well as for the DEPOSIT MM command. The console will have loaded the
:6096     address into LS 5 and the data into LS 6 before executing this routine.
:6097     The routine issues a memory request with the memory function=WRITE.P
:6098     and the data type=longword. It then issues a WRITE.MEM LS[6] instruc-
:6099     tion to write the data into memory. A check on ERROR SUM is made to
:6100     assure that the write occurred without an error. If ERROR SUM is as-
:6101     serted, the CPU will issue a CPU ACK before asserting CPU ATTN to in-
:6102     form the console that an error occurred.
:6103
:6104 .CALLING SEQUENCE:
:6105
:6106     The console loads the address of a pointer to this routine (a JMP in-
:6107     struction at WCS location 54) into the UPC and starts the CPU.
:6108
:6109 .INPUT PARAMETERS:
:6110
:6111     LS 5 - Main memory physical address
:6112     LS 6 - Main memory longword data
:6113
:6114 .IMPLICIT INPUTS:
:6115
:6116     None
:6117
:6118 .OUTPUT PARAMATERS:
:6119
:6120     Main Memory address specified is written with data from LS 6.
:6121
:6122 .IMPLICIT OUTPUTS:
:6123
:6124     A memory error asserts CPU ACK to inform the console of a write error.
:6125
:6126 .SIDE EFFECTS:
:6127
:6128     None
:6129
:6130 .---
:6131
:6132 .DEPOSIT.MM:
:6133     MEM.REQ[WRITE.P] ADRS[T5.SUB] DT[LONG] ;SET UP FOR WRITE TO MAIN
:6134     ; MEMORY USING THE PHYSICAL ADDRESS
:6135     ; STORED IN LS 5
:6136     WRITE.MEM LS[T6.SUB], ;WRITE THE DATA FROM LS 6 INTO MAIN
:6137     ; MEMORY
:6138     SKIP.IF[MEM.REF.OK] ;SKIP NEXT INSTRUCTION IF NO MEMORY
:6139     ; ERROR (NO ERR SUM)
:6140     MISC [SET.CP.ACK] ;SET CPU ACK IF ERROR SUM ASSERTED
:6141
  
```

U 65C, 990A,75

U 65D, 320C,1D

U 65E, 10D0,15

ZZ-ENKCH-1.0	:	ENKCH.MIC	DEPOSIT MAIN MEMORY	I 11	19-MAY-83	Fiche 1	Frame I11	Sequence 138
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module Rev 01.00		
:	:	ENKCH.MIC	DEPOSIT MAIN MEMORY ROUTINE			Page 135		

U 65F, 3644,15	:	6142	MOV LS[4] TO WR[0]	:PUT A 4 IN WR 0
U 660, 6COA,15	:	6143	ADD WR[0] TO LS[15.SUB]	:INCREMENT LS 5 FOR NEXT LONGWORD DEP
U 661, 8868,74	:	6144	JMP [WAIT.SUB]	:JUMP TO SET ATTN AND WAIT

ZZ-ENKCH-1.0 :
: ENKCH.MCR
: ENKCH.MIC

ENKCH.MIC

MICRO2 1M(01)
EXAMINE MAIN MEMORY ROUTINE

EXAMINE MAIN MEMORY ROUTINE
19-MAY-83 13:28:52

ENKCH Micro-diagnostic for IDC module

Fiche 1 Frame J11

Sequence 139
Rev 01.00

Page 136

:6145 :PAGE " EXAMINE MAIN MEMORY ROUTINE"
:6146 :+++

:6147 :
:6148 :FUNCTIONAL DESCRIPTION:
:6149 :

:6150 : This routine is used to read a longword from main memory at the Phy-
:6151 : sical address specified. The address need not be on a longword bound-
:6152 : ary. This routine is used by the console on an EXAMINE.MM command.
:6153 : The console will have loaded LS 5 with the physical address specified
:6154 : before executing this routine.

:6155 : The routine will first write CSR 1 to set the INH REP CRD (inhibit
:6156 : report correctable read data) bit to prevent an error sum from occurring
:6157 : on a single bit error which has been corrected. Then the routine is-
:6158 : sues a memory request with memory function=READ.P and DT=longword. It
:6159 : then reads the memory location and puts the result in LS 6. It also
:6160 : checks for an ERROR SUM and issues a CPU ACK if an error occurred (sin-
:6161 : gle bit errors that have been corrected will not cause an ERR SUM). If
:6162 : an error occurred, CPU ACK will be set before issuing CPU ATTN to in-
:6163 : form the console that an error occurred.

:6164 :
:6165 :CALLING SEQUENCE:
:6166 :

:6167 : The console loads the address of a pointer to this routine (a JMP in-
:6168 : struction at WCS location 55) into the UPC and starts the CPU.

:6169 :
:6170 :INPUT PARAMETERS:
:6171 :

:6172 : LS 5 - Physical address in main memory to read.

:6173 :
:6174 :IMPLICIT INPUTS:
:6175 :

:6176 : LS[INH.CRD] - Data to write into CSR 1 to inhibit error sum on a CRD.
:6177 : LS[CSR1] - Address for writing CSR 1.

:6178 :
:6179 :OUTPUT PARAMATERS:
:6180 :

:6181 : LS 6 - Longword data from physical memory address specified.

:6182 :
:6183 :IMPLICIT OUTPUTS:
:6184 :

:6185 : CPU ACK if an error sum occurs.

:6186 :
:6187 :SIDE EFFECTS:
:6188 :

:6189 : None

:6190 :
:6191 :---
:6192 :

:6193 :EXAMINE.MM:

U 662, 1D45,F5 :6194 : MEM.REQ[WRITE.CSR] ADRS[CSR1] DT[LONG] ;ISSUE MEMORY REQUEST TO

:6195 : : WRITE CSR 1

U 663, B278,15 :6196 : WRITE.MEM LS[INH.CRD] :SET INH REP CRD IN CSR 1 AND CLEAR

:6197 : : ECC DISABLE

U 664, 180B,F5 :6198 : MEM.REQ[READ.P] ADRS[TS.SUB] DT[LONG] ;ISSUE MEMORY REQUEST TO
:6199 :

ZZ-ENKCH-1.0	:	ENKCH.MIC	EXAMINE MAIN MEMORY	K 11	Fiche 1	Frame K11	Sequence 140
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00
:	:	ENKCH.MIC	EXAMINE MAIN MEMORY ROUTINE				Page 137

U 665, 380C,1D	:6200		: READ LONGWORD DATA FROM MAIN MEMORY
	:6201		: PHYSICAL ADDRESS IN LS 5
	:6202	MOV MEM.DATA TO LS[T6.SUB],	: READ DATA AND PUT IN LS 6
	:6203	SKIP.IF[MEM.REF.OK]	: SKIP NEXT INSTRUCTION IF NO MEMORY
	:6204		: ERROR (NO ERROR SUM)
U 666, 10D0,15	:6205	MISC [SET.CP.ACK]	: SET CPU ACK IF AN ERROR OCCURRED
U 667, 8868,74	:6206	JMP [WAIT.SUB]	: JUMP TO SET ATTN AND WAIT

:6207 .page " EXAMINE IDC ROUTINES"
 :6208 :
 :6209 :

:6210 FUNCTIONAL DESCRIPTION:
 :6211 :

:6212 The following routines are used to read data from the optional IDC
 :6213 module's registers. The result is put in LS 6 for the console to
 :6214 print out.
 :6215 :

:6216 CALLING SEQUENCE:
 :6217 :

:6218 The console loads the address of a pointer to this routine (a JMP in-
 :6219 struction at WCS location 5A through 5E) into the UPC and starts the
 :6220 CPU.
 :6221 :

:6222 INPUT PARAMETERS:
 :6223 :

:6224 None
 :6225 :

:6226 IMPLICIT INPUTS:
 :6227 :

:6228 None
 :6229 :

:6230 OUTPUT PARAMATERS:
 :6231 :

:6232 LS 6 - Data from IDC register specified.
 :6233 :

:6234 IMPLICIT OUTPUTS:
 :6235 :

:6236 None
 :6237 :

:6238 SIDE EFFECTS:
 :6239 :

:6240 None
 :6241 :

:6242 :---
 :6243 :

:6244 EXAMINE.ICSR:
 :6245 :

:6245 MISC [SET.PORT.I.0] ;SELECT PORT TO BUS
 :6246 PORT.READ PORT.ADRS[CSR] ;SEND THE READ COMMAND
 :6247 JMP [READ.IDC] ;READ THE DATA
 :6248 :

:6249 EXAMINE.IDAR:
 :6250 :

:6250 MISC [SET.PORT.I.0] ;SELECT PORT TO BUS
 :6251 PORT.READ PORT.ADRS[DAR] ;SEND THE READ COMMAND
 :6252 JMP [READ.IDC] ;READ THE DATA
 :6253 :

:6254 EXAMINE.DBUF:
 :6255 :

:6255 MISC [SET.PORT.I.0] ;SELECT PORT TO BUS
 :6256 PORT.READ PORT.ADRS[DATA.WORD] ;SEND THE READ COMMAND
 :6257 JMP [READ.IDC] ;READ THE DATA
 :6258 :

:6259 EXAMINE.PATT:
 :6260 :

:6260 MISC [SET.PORT.I.0] ;SELECT PORT TO BUS
 :6261 PORT.READ PORT.ADRS[PATTERN] ;SEND THE READ COMMAND

U 668, 9090,15
 U 669, 9780,15
 U 66A, 0867,64

U 66B, 9090,15
 U 66C, 1784,15
 U 66D, 0867,64

U 66E, 9090,15
 U 66F, 9780,15
 U 670, 0867,64

U 671, 9090,15
 U 672, 1790,15

ZZ-ENKCH-1.0	:	ENKCH.MIC	EXAMINE IDC ROUTINE	M 11	Fiche 1	Frame M11	Sequence 142
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00
:	:	ENKCH.MIC	EXAMINE IDC ROUTINES				Page 139

U 673, 0867.64	:	6262	JMP [READ.IDC]	;READ THE DATA
	:	6263		
	:	6264	EXAMINE.POSIT:	
U 674, 9090.15	:	6265	MISC [SET.PORT.I.0]	;SELECT PORT TO BUS
U 675, 9794.15	:	6266	PORT.READ PORT.ADRS[POSITION]	;SEND THE READ COMMAND
	:	6267		
	:	6268		
	:	6269	READ.IDC:	
U 676, 3A0C.15	:	6270	MOV PORT TO LS[T6.SUB]	;READ DATA AND PUT IN LS 6
U 677, 1080.15	:	6271	MISC [SET.ACC.I.0]	;RETURN SELECT TO ACCELERATOR
U 678, 8868.74	:	6272	JMP [WAIT.SUB]	;JUMP TO SET ATTN AND WAIT

:6273 .page " DEPOSIT IDC ROUTINES"
 :6274 :+++

:6275 :
 :6276 : FUNCTIONAL DESCRIPTION:
 :6277 :

:6278 : The following routines are used to write data to the optional IDC
 :6279 : module's registers. The data is taken from LS 6 which is previously
 :6280 : loaded by the monitor when the DEPOSIT command is executed.
 :6281 :

:6282 : CALLING SEQUENCE:
 :6283 :

:6284 : The console loads the address of a pointer to this routine (a JMP in-
 :6285 : struction at WCS location 5F through 61) into the UPC and starts the
 :6286 : CPU.
 :6287 :

:6288 : INPUT PARAMETERS:
 :6289 :

:6290 : LS 6 - Data pattern to write.
 :6291 :

:6292 : IMPLICIT INPUTS:
 :6293 :

:6294 : None
 :6295 :

:6296 : OUTPUT PARAMATERS:
 :6297 :

:6298 : None
 :6299 :

:6300 : IMPLICIT OUTPUTS:
 :6301 :

:6302 : None
 :6303 :

:6304 : SIDE EFFECTS:
 :6305 :

:6306 : None
 :6307 :

:6308 : ---
 :6309 :

:6310 : DEPOSIT.ICSR:
 :6311 :

:6311 : MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
 :6312 : PORT.WRITE PORT.ADRS[CSR] WITH WR[0] ;WRITE TO IDC
 :6313 : JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
 :6314 :

:6315 : DEPOSIT.IDAR:
 :6316 :

:6316 : MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
 :6317 : PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE TO IDC
 :6318 : JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
 :6319 :

:6320 : DEPOSIT.DBUF:
 :6321 :

:6321 : MOV LS[T6.SUB] TO WR[0] ;GET DATA PATTERN TO WRITE
 :6322 : PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0] ;WRITE TO IDC
 :6323 : JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
 :6324 :
 :6325 :
 :6326 :
 :6327 :

U 679, 360C,15
 U 67A, 17A0,15
 U 67B, 8868,74

U 67C, 360C,15
 U 67D, 97A4,15
 U 67E, 8868,74

U 67F, 360C,15
 U 680, 17AC,15
 U 681, 8868,74

ZZ-ENKCH-1.0	:	ENKCH.MIC	DEPOSIT IDC ROUTINE	B 12	Fiche 1	Frame B12	Sequence 144
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00
:	:	ENKCH.MIC	DEPOSIT IDC ROUTINES				Page 141

U 682, 17C0,15	:6328	CLEAR.FIFO.ADDR:		
U 683, 8868,74	:6329	PORT.CONTROL [CLEAR.FIFO.CNTR]	:	CLEAR ADDRESS IN FIFO A OR FIFO B
	:6330	JMP [WAIT.SUB]	:	JUMP TO SET ATTN AND WAIT
	:6331			
	:6332			
U 684, 17D8,15	:6333	SELECT.FIFO.A:		
U 685, 8868,74	:6334	PORT.CONTROL [SEL.FIFO.A]	:	SELECT FIFO A
	:6335	JMP [WAIT.SUB]	:	JUMP TO SET ATTN AND WAIT
	:6336			
	:6337			
U 686, 97DC,15	:6338	SELECT.FIFO.B:		
	:6339	PORT.CONTROL [SEL.FIFO.B]	:	SELECT FIFO B
	:6340			
	:6341	WAIT.SUB:		
U 687, 10E0,15	:6342	MISC [SET.CP.ATTN]	:	ISSUE CPU ATTN TO CONSOLE
	:6343	WAIT.DE.EX:		
U 688, 8868,84	:6344	JMP [WAIT.DE.EX]	:	LOOP SELF UNTIL CONSOLE TAKES CONTROL

:6345 .PAGE " SAVE WR ROUTINE"

:6346 :+++

:6347 :
:6348 :FUNCTIONAL DESCRIPTION:

:6349 :
:6350 :This routine saves WRs 0-3 in LS locations 1-4. When the console takes
:6351 :control of the CPU, it calls this routine so that the working registers
:6352 :can be used by the console. The WRs are restored by another routine
:6353 :called by the console before the CPU resumes execution (on a COninue
:6354 :command, for example).

:6355 :This routine simply moves data from WR 0-3 to LS locations 1-4 and
:6356 :then issues a CPU ATTN to the console. It then loops on a JMP self
:6357 :until the console takes over.

:6358 :
:6359 :CALLING SEQUENCE:

:6360 :
:6361 :The console loads the address of a pointer to this routine (a JMP in-
:6362 :struction at WCS location 46) into the UPC and starts the CPU.

:6363 :
:6364 :INPUT PARAMETERS:

:6365 :
:6366 :None

:6367 :
:6368 :IMPLICIT INPUTS:

:6369 :
:6370 :None

:6371 :
:6372 :OUTPUT PARAMATERS:

:6373 :
:6374 :LS 1 - Contents of WR 0
:6375 :LS 2 - Contents of WR 1
:6376 :LS 3 - Contents of WR 2
:6377 :LS 4 - Contents of WR 3

:6378 :
:6379 :IMPLICIT OUTPUTS:

:6380 :
:6381 :None

:6382 :
:6383 :SIDE EFFECTS:

:6384 :
:6385 :None

:6386 :
:6387 :---

:6388 :
:6389 :SAVE.WR:

U 689, 3E02,15	:6390		
U 68A, BE04,95	:6391	MOV WR[0] TO LS[SAV.WR0]	:SAVE WR 0 IN LS LOCATION 1
U 68B, 3E07,15	:6392	MOV WR[1] TO LS[SAV.WR1]	:SAVE WR 1 IN LS LOCATION 2
U 68C, 3E09,95	:6393	MOV WR[2] TO LS[SAV.WR2]	:SAVE WR 2 IN LS LOCATION 3
U 68D, 8868,74	:6394	MOV WR[3] TO LS[SAV.WR3]	:SAVE WR 3 IN LS LOCATION 4
	:6395	JMP [WAIT.SUB]	:JUMP TO SET ATTN AND WAIT

```

:6396 .PAGE  " RESTORE WR ROUTINE"
:6397 .+++
:6398
:6399 .FUNCTIONAL DESCRIPTION:
:6400
:6401     This routine restores the 2901 working registers save in LS 1-4. The
:6402     console calls this routine before the CPU is restarted after it has
:6403     been halted by the console.
:6404     The routine simply moves the data from LS locations 1-4 to WRs 0-3
:6405     and then issues a CPU ATTN to the console. A JMP self is executed next
:6406     waiting for the console to take control.
:6407
:6408 .CALLING SEQUENCE:
:6409
:6410     The console loads the address of a pointer to this routine (a JMP in-
:6411     struction at WCS location 47) into the UPC and starts the CPU.
:6412
:6413 .INPUT PARAMETERS:
:6414
:6415     LS 1 contains the data to be restored in WR 0.
:6416     LS 2 contains the data to be restored in WR 1.
:6417     LS 3 contains the data to be restored in WR 2.
:6418     LS 4 contains the data to be restored in WR 3.
:6419
:6420 .IMPLICIT INPUTS:
:6421
:6422     None
:6423
:6424 .OUTPUT PARAMATERS:
:6425
:6426     WR 0 gets data from LS 1.
:6427     WR 1 gets data from LS 2.
:6428     WR 2 gets data from LS 3.
:6429     WR 3 gets data from LS 4.
:6430
:6431 .IMPLICIT OUTPUTS:
:6432
:6433     None
:6434
:6435 .SIDE EFFECTS:
:6436
:6437     None
:6438
:6439 .---
:6440
:6441 .RESTORE.WR:
:6442     MOV LS[SAV.WR0] TO WR[0]      ;RESTORE WR 0
:6443     MOV LS[SAV.WR1] TO WR[1]      ;RESTORE WR 1
:6444     MOV LS[SAV.WR2] TO WR[2]      ;RESTORE WR 2
:6445     MOV LS[SAV.WR3] TO WR[3]      ;RESTORE WR 3
:6446     JMP [WAIT.SUB]                ;JUMP TO SET ATTN AND WAIT
  
```

```

U 68E, B602,15
U 68F, 3604,95
U 690, B607,15
U 691, B609,95
U 692, 8868,74
  
```



```

:6447 .PAGE 'WCS SUBROUTINES FOR CPU USE'
:6448 .TOC ' ERROR ROUTINE'
:6449 .+++
:6450
:6451 FUNCTIONAL DESCRIPTION:
:6452
:6453 This routine is used to detect and report diagnostic errors occurring
:6454 during WCS based micro-diagnostics. The WCS micro-code sets up the
:6455 parameters listed below and calls this routine. The routine compares
:6456 WR 1 (expected data) and WR 0 (received data) according to the error
:6457 mask. Bits set in the mask indicate bits that are not checked for
:6458 errors.
:6459 If WR 0 and WR 1 are equal (no error) the routine executes a return+1
:6460 to the calling point. The only exception to this is if loop on error
:6461 is set. In that case, the error number is checked to see if that error
:6462 occurred previously. If so, then the error loop is forced by doing a
:6463 return. This will lock an error loop in on intermittent errors.
:6464 If WR 0 and WR 1 are not equal (an error) then bit 8 is set in the
:6465 CONTROL AND STATUS word, expected and received data is set up in LS for
:6466 printout, and flags are checked in the ERROR CONTROL word. This con-
:6467 trols whether printouts are disabled, loop on error is enabled, etc.
:6468 After printing the error, a return+1 is made to the calling point.
:6469 Loop on error causes a return rather than return+1 to cause an error
:6470 loop. Also CPU ACKNOWLEDGE is toggled at the end of the routine for
:6471 a scope sync point.
:6472
:6473 CALLING SEQUENCE:
:6474 JSR [CHECK.RESULT]
:6475
:6476 INPUT PARAMETERS:
:6477
:6478 WR[0] = Received data i.e. the actual result of the test
:6479 WR[1] = Expected data i.e. what the result should have been
:6480
:6481 IMPLICIT INPUTS:
:6482
:6483 LS[ERROR.MASK] = Used to mask out bits that are not to be tested
:6484 LS[ERROR.NUMBER] = Current error number
:6485 LS[PREVIOUS.ERROR] = Error number of previous data
:6486 LS[CONTROL.STATUS] = Control and Status word. Contains information
:6487 written by the CPU to inform the monitor what to do.
:6488 LS[ERROR.CONTROL] = Information such as loop-on-error, bell-on-error,
:6489 no error report, and halt on error.
:6490
:6491 OUTPUT PARAMETERS:
:6492
:6493 NONE
:6494
:6495 IMPLICIT OUTPUTS:
:6496
:6497 LS[CONTROL.STATUS] = Control and Status with new status information
:6498 LS[PREVIOUS.ERROR] = Last error number (modified only if error and
:6499 loop on error is enabled)
:6500 LS[EXPECTED.DATA] = Expected result if an error was detected
:6501

```

```

:6502      LS[RECEIVED.DATA] = Actual result if an error was detected
:6503
:6504      SIDE EFFECTS:
:6505
:6506      WR[0], WR[1], and the Q reg are modified and are not restored before
:6507      returning.
:6508      If loop-on-error is enabled and an error loop is to be executed, CPU
:6509      ACKNOWLEDGE will be toggled for a scope sync.
:6510
:6511      ---
:6512
:6513      CHECK.RESULT:
U 693, 458A,15 :6514      BIC LS[ERROR.MASK] TO WR[0]      ;MASK OFF NOT TESTED BITS (CLEAR THEM)
:6515      ; FOR RECEIVED DATA
U 694, C58A,95 :6516      BIC LS[ERROR.MASK] TO WR[1]      ;DITTO FOR EXPECTED DATA
:6517
:6518      XOR WR[0] WITH WR[1] TO Q,      ;CMP RECEIVED DATA IN WR[0] WITH
U 695, AB80,B5 :6519      DT(LONG)&SET.ALU.CC      ; EXPECTED DATA IN WR[1] FOR ERROR
U 696, 8869,F1 :6520      JMP.IF[NEQ] TO [ERROR]      ;JUMP IF ERROR
:6521
:6522      MOV LS[ERROR.CONTROL] TO WR[0] ;GET ERROR CONTROL WORD FROM LS
:6523      BIT WR[0] WITH LS[LOE],      ;SEE IF LOOP ON ERROR IS ENABLED
U 698, 5940,35 :6524      DT(LONG)&SET.ALU.CC      ; SET CONDITION CODES
U 699, DB00,01 :6525      SKIP.IF[BIT.SET]      ;SKIP IF IT IS
U 69A, DB00,16 :6526      RETURN+1      ;ELSE RETURN+1 (NO ERROR AND NO LOOP)
:6527
:6528      MOV LS[PREVIOUS.ERROR] TO WR[0] ;IF NO ERROR BUT LOOP ON ERROR IS
:6529      ; ENABLED, SEE IF THERE WAS A
:6530      ; PREVIOUS ERROR
U 69C, CD82,35 :6531      XOR WR[0] WITH LS[ERROR.NUMBER] TO Q,
:6532      DT(LONG)&SET.ALU.CC      ;IS THIS THE SAME SPOT THAT HAD AN
:6533      ; ERROR BEFORE? (INTERMITTANT ERROR)
U 69D, 086B,91 :6534      JMP.IF[NEQ] TO [RET+1]      ;JUMP IF NOT TO RETURN WITHOUT ERROR
:6535      ; LOOPING (RETURN+1)
U 69E, 086B,C4 :6536      JMP [RET]      ;ELSE CONTINUE TO LOOP (WE WILL LOOP
:6537      ; ON ERROR EVEN IF IT GOES AWAY)
:6538
:6539      ERROR:
U 69F, 3E86,15 :6540      MOV WR[0] TO LS[RECEIVED.DATA] ;PUT RESULT OF TEST IN LS FOR PRINTOUT
U 6A0, 3690,15 :6541      MOV LS[ERROR.CONTROL] TO WR[0] ;GET ERROR CONROL BITS TO CHECK FOR
:6542      ; LOOP COMMAND
:6543      BIT WR[0] WITH LS[LOOP.COMMAND], ;SEE IF BIT 6 SET
U 6A1, 594C,35 :6544      DT(LONG)&SET.ALU.CC      ; SET CONDITION CODES
U 6A2, 086B,C1 :6545      JMP.IF[BIT.SET] TO [RET]      ;GO LOOP IF SET (FAST LOOP)
:6546
:6547      MOV WR[1] TO LS[EXPECTED.DATA] ;PUT EXPECTED DATA IN LS FOR PRINTOUT
:6548
:6549      MOV LS[CONTROL.STATUS] TO WR[1] ;GET CONTROL AND STATUS WORD FROM LS
U 6A4, 3680,95 :6550      BIC LS[#FE7FFFFF] TO WR[1]      ;CLEAR ALL BUT BITS 23&24 IN CONTROL
U 6A5, C532,95 :6551      ; AND STATUS WORD
:6552      BIS LS[ERROR] TO WR[1]      ;SET BIT 8 IN CONTROL AND STATUS WORD
:6553      ; (INDICATES AN ERROR WAS DETECTED)
U 6A6, C750,95 :6554      MOV WR[1] TO LS[CONTROL.STATUS] ;WRITE UPDATED CONTROL AND STATUS WORD
:6555      ; BACK TO LS[40]
:6556

```

ZZ-ENKCH-1.0	:	ENKCH.MIC	ERROR ROUTINE	G 12	Fiche 1	Frame G12	Sequence 149
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83	ENKCH Micro-diagnostic for IDC module		Rev 01.00
:	:	ENKCH.MIC	ERROR ROUTINE	13:28:52			Page 146

U 6A8, D944,35	:6557	BIT WR[0] WITH LS[BELL],	:SEE IF BELL ON ERROR IS SET (WR 0 STILL
U 6A9, 886A,D9	:6558		:CONTAINS ERROR CONTROL WORD)
	:6559	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 6AA, 10E0,15	:6560	JMP.IF[BIT.CLR] TO [CHECK.NER]	:IF BELL ON ERROR IS NOT SET, JUMP TO
	:6561		:SEE IF ERROR REPORTING IS INHIBITED
U 6AB, 086A,B4	:6562	MISC [SET.CP.ATTN]	:ELSE SET CPU ATTN (RING MY BELL)
U 6AC, 086B,64	:6563		
	:6564	WAIT.1: JMP [WAIT.1]	:WAIT FOR CONSOLE TO STOP ME
	:6565	JMP [LOOP]	:GO TO CHECK LOOP-ON-ERROR AFTER
	:6566		:CONSOLE HAS FINISHED
	:6567		
	:6568	CHECK.NER:	
	:6569	BIT WR[0] WITH LS[NER],	:IF NO BELL SEE IF ERROR REPORT IS
U 6AD, D942,35	:6570		:INHIBITED
U 6AE, 886B,21	:6571	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
	:6572	JMP.IF[BIT.SET] TO [CHECK.HALT]	:JUMP IF ERROR REPORT IS INHIBITED TO
	:6573		:CHECK FOR HALT ON ERROR
	:6574		
U 6AF, 10E0,15	:6575	MISC [SET.CP.ATTN]	:SET CPU ATTN (REPORT ERROR)
U 6B0, 086B,04	:6576	WAIT.2: JMP [WAIT.2]	:WAIT FOR CONSOLE TO STOP ME
U 6B1, 086B,64	:6577	JMP [LOOP]	:GO TO CHECK LOOP-ON-ERROR AFTER
	:6578		:CONSOLE HAS FINISHED
	:6579		
	:6580	CHECK.HALT:	
U 6B2, 5946,35	:6581	BIT WR[0] WITH LS[HOE],	:SEE IF HALT ON ERROR IS ENABLED
U 6B3, 886B,69	:6582	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
	:6583	JMP.IF[BIT.CLR] TO [LOOP]	:JUMP IF NOT TO CHECK LOOP-ON-ERROR
	:6584		
U 6B4, 10E0,15	:6585	MISC [SET.CP.ATTN]	:ELSE SET CPU ATTN (HALT ON ERROR)
U 6B5, 086B,54	:6586	WAIT.3: JMP [WAIT.3]	:WAIT FOR CONSOLE TO STOP ME
	:6587		
U 6B6, 3690,15	:6588	LOOP: MOV LS[ERROR.CONTROL] TO WR[0]	:GET ERROR CONTROL BITS (NER, HOE ETC.)
	:6589	BIT WR[0] WITH LS[LOE],	:SEE IF LOOP-ON-ERROR
U 6B7, 5940,35	:6590	DT(LONG)&SET.ALU.CC	:SET CONDITION CODES
U 6B8, DB00,01	:6591	SKIP.IF[BIT.SET]	:SKIP IF LOOP ON ERROR IS ENABLED
	:6592		
U 6B9, DB00,16	:6593	RET+1: RETURN+1	:ELSE RETURN+1 FOR NO ERROR LOOP
	:6594		
U 6BA, 3682,15	:6595	MOV LS[ERROR.NUMBER] TO WR[0]	:GET ERROR NUMBER IF LOOPING
	:6596		
U 6BB, BEA0,15	:6597	MOV WR[0] TO LS[PREVIOUS.ERROR]	:AND SAVE IT IN PREVIOUS ERROR
	:6598		
U 6BC, 10D0,15	:6599	RET: MISC [SET.CP.ACK]	:SET CPU ACKNOWLEDGE FOR SCOPE SYNC
	:6600	MISC [CLR.CP.ATTN.AND.ACK],	:AND THEN CLEAR IT
U 6BD, 10C0,14	:6601	RETURN	:RETURN TO TEST AND LOOP ON ERROR
	:6602		


```

:6603 .PAGE
:6604
:6605
:6606
:6607 ERR.NO.TEST:
U 6BE, DB00,15 :6608 NOP ;NOP WILL CAUSE A SEQUENCE ERROR
U 6BF, B65E,15 :6609 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR CONTROL AND
:6610 ;STATUS WORD
U 6C0, 3E80,15 :6611 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
:6612 ;BIT 15 INDICATES START OF TEST TO
:6613 ;CONSOLE
U 6C1, 8868,74 :6614 JMP [WAIT.SUB] ;JUMP TO SET ATTN AND WAIT
:6615 .TOC "START TRANSFER ROUTINE"
:6616
:6617 This routine loads the LS with information necessary for these tests
:6618 to run. It will then issue a CPU ATTN with the control and status word
:6619 clear to indicate that all transfers are complete.
:6620
:6621 RESULT: LS LOADED WITH CONSTANTS. MONITOR INFORMED TRANSFERS OVER.
:6622
:6623 ---
:6624 TRANSFER.POINT:
:6625
:6626
U 6C2, 2F80,15 :6627 CLR WR[0] ;START WITH 0 IN WR[0]
U 6C3, BFFE,15 :6628 MOV WR[0] TO LS[PSL.HW] ;MAKE SURE COMPAT MODE IS CLEAR
:6629
U 6C4, 2040,15 :6630 INC WR[0] ;NOW HAVE 1 IN WR[0]
U 6C5, 23D0,15 :6631 ASHL WR[0] ;10(B)
U 6C6, 2040,15 :6632 INC WR[0] ;11(B)
U 6C7, 23D0,15 :6633 ASHL WR[0] ;110(B)
U 6C8, 2040,15 :6634 INC WR[0] ;111(B)
U 6C9, 23D0,15 :6635 ASHL WR[0] ;1110(B)
U 6CA, 23D0,15 :6636 ASHL WR[0] ;1 1100(B)
U 6CB, 23D0,15 :6637 ASHL WR[0] ;11 1000(B)
U 6CC, 23D0,15 :6638 ASHL WR[0] ;111 0000(B) NOW HAVE 70(H) IN WR 0.
:6639 ;THIS IS CORRECT START ADDRESS OF DATA
:6640 ;SECTION (INCLUDING LONGWORD COUNT,
:6641 ;DESTINATION, AND DESTINATION ADDRESS)
U 6CD, 3E0E,15 :6642 MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD LS 7 WITH POINTER TO TRANSFER
:6643 ;DATA SECTION.
:6644
U 6CE, 2F80,15 :6645 CLR WR[0] ;0 IN WR 0
U 6CF, 2040,15 :6646 INC WR[0] ;1 IN WR 0
U 6D0, 23D0,15 :6647 ASHL WR[0] ;10(B)
U 6D1, 23D0,15 :6648 ASHL WR[0] ;100(B)
U 6D2, 23D0,15 :6649 ASHL WR[0] ;1000(B)
U 6D3, 23D0,15 :6650 ASHL WR[0] ;1 0000(B)
U 6D4, 23D0,15 :6651 ASHL WR[0] ;10 0000(B)
U 6D5, 23D0,15 :6652 ASHL WR[0] ;100 0000(B)
U 6D6, 23D0,15 :6653 ASHL WR[0] ;1000 0000(B)
U 6D7, 23D0,15 :6654 ASHL WR[0] ;1 0000 0000(B)
U 6D8, 23D0,15 :6655 ASHL WR[0] ;10 0000 0000(B)
U 6D9, 23D0,15 :6656 ASHL WR[0] ;100 0000 0000(B)
U 6DA, 23D0,15 :6657 ASHL WR[0] ;1000 0000 0000(B)

```

ZZ-ENKCH-1.0	:	ENKCH.MIC	START TRANSFER ROUTINE	I 12	Fiche 1 Frame I12	Sequence 151
:	:	ENKCH.MCR	MICRO2 1M(01)	19-MAY-83 13:28:52	ENKCH Micro-diagnostic for IDC module	Rev 01.00
:	:	ENKCH.MIC	START TRANSFER ROUTINE			Page 148

U 6DB, 23D0,15	:6658	ASHL WR[0]	:1 0000 0000 0000(B)
U 6DC, 23D0,15	:6659	ASHL WR[0]	:10 0000 0000 0000(B)
U 6DD, 3E80,15	:6660	MOV WR[0] TO LS[CONTROL.STATUS]	:SET BIT 13 IN LS CONTROL AND STATUS
	:6661		:WORD (BIT 13 INDICATES CONSOLE MUST
	:6662		:PERFORM A DATA TRANSFER)
U 6DE, 10E0,15	:6663	MISC [SET.CP.ATTN]	:ISSUE CPU ATTN TO CONSOLE (DO FIRST LS
	:6664		:TRANSFER TO LOCATIONS 0 THROUGH 77(H))
	:6665	WAIT.XFER1:	
U 6DF, 086D,F4	:6666	JMP [WAIT.XFER1]	:LOOP HERE WAITING FOR CONSOLE TO RE-
	:6667		:SPOND
	:6668		
U 6E0, 36CA,15	:6669	MOV LS[#162(H)] TO WR[0]	:GET START ADDRESS OF SECOND HALF OF LS
	:6670		:TRANSFER
U 6E1, 3E0E,15	:6671	MOV WR[0] TO LS[TRANSFER.POINTER]	:LOAD START ADDRESS IN LS FOR CONSOLE
U 6E2, 365A,15	:6672	MOV LS[XFER.DATA] TO WR[0]	:BIT 13 INDICATES TO DO DATA TRANSFER
U 6E3, 3E80,15	:6673	MOV WR[0] TO LS[CONTROL.STATUS]	:SET UP CONTROL AND STATUS WORD
U 6E4, 10E0,15	:6674	MISC [SET.CP.ATTN]	:ISSUE CPU ATTN TO CONSOLE (DO SECOND LS
	:6675		:TRANSFER TO LOCATIONS 80 THROUGH F7(H))
	:6676	WAIT.XFER2:	
U 6E5, 086E,54	:6677	JMP [WAIT.XFER2]	:LOOP HERE WAITING FOR CONSOLE TO RE-
	:6678		:SPOND
	:6679		
U 6E6, 9B9D,75	:6680	MEM.REQ[READ.V.RCHK.IFILL] ADRS[ZERO] DT[LONG]	:THIS INSTRUCTION USED TO FREE MEMORY
	:6681		:IF HUNG WAITING FOR A DATA RCVD.
U 6E7, 3022,15	:6682	MOV MEM.DATA TO WR[0]	:THIS INSTRUCTION USED TO FREE MEMORY
	:6683		:IF HUNG IN A OCTA FLOW
U 6E8, 3022,15	:6684	MOV MEM.DATA TO WR[0]	:NEEDS 4 OF THESE
U 6E9, 3022,15	:6685	MOV MEM.DATA TO WR[0]	
U 6EA, 3022,15	:6686	MOV MEM.DATA TO WR[0]	:DONE WITH MEMORY INITIALIZATION
	:6687		
U 6EB, 17CC,15	:6688	PORT.CONTROL [CLEAR.IDC]	:INITIALIZE IDC TO IDLE LOOP
U 6EC, 17CC,15	:6689	PORT.CONTROL [CLEAR.IDC]	
U 6ED, 17CC,15	:6690	PORT.CONTROL [CLEAR.IDC]	
	:6691		
	:6692		
U 6EE, 17D4,15	:6693	PORT.CONTROL [CLEAR.AUTO]	:CLEAR AUTOMODE
	:6694		
U 6EF, 3678,15	:6695	MOV LS[BIT28] TO WR[0]	
U 6F0, B64E,95	:6696	MOV LS[BIT7] TO WR[1]	
U 6F1, AEC2,15	:6697	BIS WR[1] TO WR[0]	
U 6F2, 17A0,15	:6698	PORT.WRITE PORT.ADRS[CSR] WITH WR[0]	:SET WRITE INHIBIT IN IDC CSR
	:6699		:AND SET CRDY
	:6700		
U 6F3, 3660,15	:6701	MOV LS[INTERUPT.EN] TO WR[0]	:SET BIT 16 IN WR
U 6F4, 3E80,15	:6702	MOV WR[0] TO LS[CONTROL.STATUS]	:SET BIT 16 TO CLEAR ANY INTERRUPTS
	:6703		:AND INFORM CONSOLE THAT XFERS ARE
	:6704		:DONE
U 6F5, 10E0,15	:6705	MISC [SET.CP.ATTN]	:CALL CONSOLE
	:6706	WAIT.XFER:	
U 6F6, 886F,64	:6707	JMP [WAIT.XFER]	:WAIT FOR CONSOLE TO RESPOND
U 6F7, 5B00,14	:6708	RETURN	:USED WHEN TRANSFER IS CALLED FROM
	:6709		:MICRO-DIAGNOSTIC ONLY

ZZ-ENKCH-1.0 : ENKCH.MIC J 12
: ENKCH.MCR MICRO2 1M(01) 19-MAY-83 13:28:52 FICHE 1 Frame J12 Sequence 152
: ENKCH.MIC START TRANSFER ROUTINE ENKCH Micro-diagnostic for IDC module Rev 01.00 Page 149

:6710 .PAGE
:6711 .REGION/700,3FFF
:6712


```

:6713 .PAGE "Test 1 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)"
:6714 ++
:6715
:6716 Test Description:
:6717
:6718 If this diagnostic is being run under APT and an IDC is not found
:6719 then an error will be reported.
:6720 This test will size for the IDC Module by writing and then reading
:6721 Bit <0> of the Disk Address Register (DAR). If this cannot be
:6722 accomplished successfully, then the message "IDC NOT PRESENT" will
:6723 be printed and the IDC microdiagnostic will not be executed. Otherwise
:6724 "IDC PRESENT" will be printed and test execution will continue.
:6725
:6726 Assumptions:
:6727
:6728 It is assumed that the CPU and the Y bus are fully operational.
:6729
:6730 Test Steps:
:6731
:6732 1. Move zero to WR[0].
:6733 2. Move #1 to WR [1].
:6734 3. Write a one to DAR <0>.
:6735 4. Move WR[0] to a temporary LS location to change the data on the
:6736 Y BUS.
:6737 5. Read DAR <0>.
:6738 6. Check result. If incorrect data, print message "IDC NOT PRESENT" and
:6739 go to Step 12.
:6740 7. Write a zero to DAR <0>.
:6741 8. Move WR[1] to a temporary LS location to change the data on the
:6742 Y BUS.
:6743 9. Read DAR <0>.
:6744 10. Check result.
:6745 11. If data is incorrect, print message "IDC NOT PRESENT".
:6746 12. If APT is present then force an error because IDC was not found.
:6747 13. Go to end of section.
:6748 14. Else print "IDC PRESENT" and continue IDC microdiagnostic execution.
:6749
:6750 Errors:
:6751
:6752 ERROR 1 - The diagnostic was being run under APT and no IDC was found.
:6753
:6754 Printout:
:6755 EXPECTED RECEIVED
:6756 N/A N/A
:6757
:6758 Debug:
:6759
:6760 If an IDC is not found by this test and one is known to exist, then
:6761 TEST 4 DISK ADDRESS REGISTER TEST should be run to allow looping on
:6762 error. A more detailed debug section is also available in TEST 4.
:6763
:6764 If the IDC is not found and one is known to exist then the fault could
:6765 be in one of the following:
:6766
:6767 1. The STATE REG, MISC DECODER PAL on the CPU module which generates
  
```

```

:6768 : the PORT INSTR H signal.
:6769 : 2. The A,B ADRS GEN PAL on the CPU module which generates the
:6770 : READ PORT L signal.
:6771 : 3. Bit <0> of the data bus transceiver
:6772 : 4. Bit <0> of the IDC I/O bus
:6773 : 5. Bit <0> of the DAR
:6774 : 6. The DAR control signals, WRITE DAR or SEL DAR which are decoded from
:6775 : the Port instruction through the PORT INTERFACE REG PAL, the PORT
:6776 : RD/WR DECODE PAL and the Port instruction decoder.
:6777 :
:6778 :--
:6779 :*****
:6780 :T.1:
U 700, B65E,15 :6781 MOV LS[BEGIN.TEST] TO WR[0] ;SET BIT 15 IN WR[0] FOR
:6782 ; CONTROL AND STATUS WORD
U 701, 3E80,15 :6783 MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND
:6784 ; STATUS WORD - BIT 15
:6785 ; INDICATES START OF TEST TO
:6786 ; CONSOLE
U 702, 10E0,15 :6787 MISC [SET.CP.ATTN] ;ISSUE CPU ATTN TO CONSOLE
:6788 WAIT.T1.0:
U 703, 0870,34 :6789 JMP [WAIT.T1.0] ;LOOP HERE WAITING FOR CONSOLE
:6790 ; TO RESPOND
:6791
U 704, E50E,15 :6792 CLR LS[T7.SUB] ;SET UP FOR MESSAGE PRINTOUT
U 705, B640,15 :6793 MOV LS[#1] TO WR[0]
U 706, BE82,15 :6794 MOV WR[0] TO LS[ERROR.NUMBER] ;SET ERROR NUMBER =1
:6795 ;*****
:6796 U 707, 3776,15 :6796 MOV LS[DAR.MASK] TO WR[0] ;CHANGE MAR 7,83 PETER GREEN
:6797 U 708, C566,15 :6797 BIC LS[BIT19] TO WR[0] ;MANUFACT PROB W/SLOW CLOCK
:6798 U 709, 4568,15 :6798 BIC LS[BIT20] TO WR[0] ;IDC NOT FOUND (INTERMITTANT)
:6799 U 70A, 3E8A,15 :6799 MOV WR[0] TO LS[ERROR.MASK] ;BITS 25 & 27 BEING SET
:6800 ;SET UP ERROR MASK TO CHECK
:6801 ; BITS <20:0> IDAR
:6802 ;*****
:6803 LOOP.T1.1:
:6804 WRITE1.DAR.T1:
U 70B, B640,15 :6805 MOV LS[#1] TO WR[0] ;GET DATA TO WRITE TO DAR
U 70C, 97A4,15 :6806 PORT.WRITE PORT.ADRS[DAR] WITH WR[0] ;WRITE A ONE TO IDC DISK
:6807 ; ADDRESS REGISTER
:6808
U 70D, 369C,15 :6809 MOV LS[ZERO] TO WR[0] ;CHANGE THE DATA IN WR[0]
U 70E, 3F32,95 :6810 MOV WR[1] TO LS[PORT0] ; AND IN LS[PORT0]
:6811
U 70F, 9090,15 :6812 MISC [SET.PORT.I.0] ;SELECT THE PORT TO BUS
U 710, 1784,15 :6813 PORT.READ PORT.ADRS[DAR] ;READ BACK THE IDC DAR
U 711, 3B32,15 :6814 MOV PORT TO LS[PORT0] ;MOV THE RECEIVED DATA TO LS
U 712, 3732,15 :6815 MOV LS[PORT0] TO WR[0] ;MOV RECEIVED DATA FOR COMPARE
U 713, 3640,95 :6816 MOV LS[#1] TO WR[1] ;GET EXPECTED DATA
:6817
U 714, 458A,15 :6818 BIC LS[ERROR.MASK] TO WR[0] ;CHANGE 3/15 P. G.
:6819 ;MASK BITS 20-31 FIX
U 715, C58A,95 :6820 BIC LS[ERROR.MASK] TO WR[1] ;CLOCK MARGINS
:6821 ;
:6822 CMP WR[0] WITH WR[1], ;CMP THE EXP AND REC DATA

```

ZZ-ENKCH-1.0 :		ENKCH.MIC		M 12		Fiche 1		Frame M12		Sequence 155	
: ENKCH.MCR		MICRO2 1M(01)		Test 1 IDC SIZING 19-MAY-83		19-MAY-83 13:28:52		ENKCH Micro-diagnostic for IDC module		Rev 01.00	
: ENKCH.MIC		Test 1		IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)						Page 152	

U 716, 2640.B5	:6823	DT(LONG)&SET.ALU.CC		
U 717, 8872.51	:6824	JMP.IF[NEQ] TO [NO.IDC.T1]		;GO PRINT NO IDC
	:6825			
	:6826	WRITE0.DAR.T1:		
U 718, 369C.15	:6827	MOV LS[#0] TO WR[0]		;GET DATA TO WRITE TO DAR
U 719, 97A4.15	:6828	PORT.WRITE PORT.ADRS[DAR] WITH WR[0]		;WRITE A ZERO TO IDC DISK
	:6829			; ADDRESS REGISTER
	:6830			
U 71A, B640.15	:6831	MOV LS[#1] TO WR[0]		;CHANGE THE DATA IN WR[0]
U 71B, 3F32.95	:6832	MOV WR[1] TO LS[PORT0]		; AND LS[PORT0]
	:6833			
U 71C, 9090.15	:6834	MISC [SET.PORT.1.0]		;SELECT THE PORT TO BUS
U 71D, 1784.15	:6835	PORT.READ PORT.ADRS[DAR]		;READ BACK THE IDC DAR
U 71E, 3B32.15	:6836	MOV PORT TO LS[PORT0]		;MOV THE RECEIVED DATA TO LS
U 71F, 3732.15	:6837	MOV LS[PORT0] TO WR[0]		;MOV RECEIVED DATA FOR COMPARE
U 720, B69C.95	:6838	MOV LS[#0] TO WR[1]		;GET EXPECTED DATA
	:6839			
U 721, 458A.15	:6840	BIC LS[ERROR.MASK] TO WR[0]		;CHANGE 3/15 P. G.
	:6841			;MASK BITS 20-31 FIX
U 722, C58A.95	:6842	BIC LS[ERROR.MASK] TO WR[1]		;CLOCK MARGINS
	:6843			
	:6844	CMP WR[0] WITH WR[1],		;CMP THE EXP AND REC DATA
U 723, 2640.B5	:6845	DT(LONG)&SET.ALU.CC		
U 724, 8873.39	:6846	JMP.IF[EQL] TO [IDC.FOUND.T1]		;GO PRINT IDC FOUND
	:6847			
	:6848	NO.IDC.T1:		
U 725, 367E.15	:6849	MOV LS[PRINT.IDC] TO WR[0]		;SET UP CONTROL STATUS WORD TO
U 726, 3E80.15	:6850	MOV WR[0] TO LS[CONTROL.STATUS]		; PRINT NO IDC FOUND
U 727, 10E0.15	:6851	MISC [SET.CP.ATTN]		;SET ATTENTION TO THE CONSOLE
	:6852	WAIT.T1.1:		
U 728, 0872.84	:6853	JMP [WAIT.T1.1]		;WAIT FOR CONSOLE TO RETURN
	:6854			; CONTROL
	:6855			
U 729, 3690.15	:6856	MOV LS[ERROR.CONTROL] TO WR[0]		;RUNNING UNDER APT?
	:6857	BIT LS[APT.PRESENT] WITH WR[0],		
U 72A, 594A.35	:6858	DT(LONG)&SET.ALU.CC		
U 72B, 0873.89	:6859	JMP.IF[BIT.S CLR] TO [END.SECTION]		;IF APT NOT PRESENT THEN DO
	:6860			; NOT RUN DIAGNOSTIC
U 72C, B66E.15	:6861	MOV LS[NA.EXP.REC] TO WR[0]		
U 72D, 3E80.15	:6862	MOV WR[0] TO LS[CONTROL.STATUS]		;SET UP TO FORCE AN ERROR
U 72E, 2F80.15	:6863	CLR WR[0]		
U 72F, 3640.95	:6864	MOV LS[#1] TO WR[1]		
U 730, 0869.3C	:6865	JSR [CHECK.RESULT]		;GO FORCE THE ERROR
U 731, 8870.84	:6866	JMP [LOOP.T1.1]		;LOOP HERE IF ERR & LOE
	:6867			
U 732, 8873.84	:6868	JMP [END.SECTION]		;NO IDC AVAILABLE, GO TO END
	:6869			; OF SECTION
	:6870			
	:6871	IDC.FOUND.T1:		
U 733, 7F0E.15	:6872	INC LS[T7.SUB]		;SET TO PRINT IDC FOUND
U 734, 367E.15	:6873	MOV LS[PRINT.IDC] TO WR[0]		;SET CONTROL STATUS WORD TO
U 735, 3E80.15	:6874	MOV WR[0] TO LS[CONTROL.STATUS]		; PRINT MESSAGE
U 736, 10E0.15	:6875	MISC [SET.CP.ATTN]		;SET ATTENTION TO THE CONSOLE
	:6876	WAIT.T1.2:		
U 737, 8873.74	:6877	JMP [WAIT.T1.2]		;WAIT FOR CONSOLE TO RETURN

ZZ-ENKCH-1.0 : ENKCH.MIC Test 1 IDC SIZING 19-MAY-83 N 12 Fiche 1 Frame N12 Sequence 156
: ENKCH.MCR MICRO2 1M(01) 19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module Rev 01.00 Page 153
: ENKCH.MIC Test 1 IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)

:6878
:6879
:6880
:6881

END.T1:

;END TEST

	:6882	.PAGE	
	:6883		
	:6884	END.SECTION:	
U 738, 365C,15	:6885	MOV LSE[BIT14] TO WR[0]	;SET UP WR 0 FOR END OF SECTION
U 739, 3E80,15	:6886	MOV WR[0] TO LSE[CONTROL.STATUS]	;SET UP CONTROL AND STATUS
	:6887		;WORD TO REPORT END OF SECTION
U 73A, 10E0,15	:6888	MISC [SET.CP.ATTN]	;ISSUE CPU ATTN TO CONSOLE
	:6889	WAIT.EOS:	
U 73B, 8873,B4	:6890	JMP [WAIT.EOS]	;LOOP HERE WAITING FOR CONSOLE
	:6891		; TO RESPOND
	:6892		
	:6893		
	:6894		
	:6895		

ZZ-ENKCH-1.0 ;
: ENKCH.MCR
:

C 13
MICRO2 1M(01) Cross Reference Listing - MAY-83
19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module
Cross Reference Listing - Field Names and Defined Values

Fiche 1 Frame C13

Sequence 158

Rev 01.00

Page 155

ZZ-ENKCH-1.0 ;
: ENKCH.MCR ;
:

Cross Reference Listing
MICRO2 1M(01) 19-MAY-83 13:28:52
Cross Reference Listing - Macro Names

D 13

19-MAY-83

Fiche 1 Frame D13

Sequence 159

ENKCH Micro-diagnostic for IDC module Rev 01.00 Page 156

ZZ-ENKCH-1.0 ;
: ENKCH.MCR
:

E 13
MICRO2 1M(01) Cross Reference Listing - Expression Names
19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module
Fiche 1 Frame E13
Sequence 160
Rev 01.00 Page 157

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 000	2441:	2442:	2443:	2444:	2445:	2446:	2447:	2448:
C 008	2449:	2450:	2451:	2452:	2453:	2454:	2455:	2456:
C 010	2460:	2461:	2462:	2463:	2464:	2465:	2466:	2467:
C 018	2468:	2469:	2470:	2471:	2472:	2473:	2474:	2475:
C 020	2484:	2485:	2486:	2487:	2488:	2489:	2490:	2491:
C 028	2492:	2493:	2494:	2495:	2496:	2497:	2498:	2499:
C 030	2500:	2501:	2502:	2503:	2504:	2505:	2506:	2507:
C 038	2508:	2509:	2510:	2511:	2512:	2513:	2514:	2515:
C 040	2524:	2525:	2526:	2527:	2528:	2529:	2530:	2531:
C 048	2532:	2533:	2534:	2535:	2536:	2537:	2538:	2539:
C 050	2540:	2541:	2542:	2543:	2544:	2545:	2546:	2547:
C 058	2548:	2549:	2550:	2551:	2552:	2553:	2554:	2555:
C 060	2563:	2564:	2565:	2566:	2567:	2568:	2569:	2570:
C 068	2571:	2572:	2573:	2574:	2575:	2576:	2577:	2578:
C 070	2579:	2580:	2581:	2582:	2583:	2584:	2585:	2586:
C 078	2587:	2588:	2589:	2590:	2591:	2592:	2593:	2594:
C 080	2601:	2603:	2605:	2607:	2610:	2612:	2614:	2616:
C 088	2619:	2621:	2623:	2625:	2628:	2630:	2632:	2634:
C 090	2637:	2639:	2641:	2643:	2646:	2648:	2650:	2652:
C 098	2655:	2657:	2659:	2661:	2664:	2666:	2668:	2670:
C 0A0	2673:	2675:	2677:	2679:	2682:	2684:	2686:	2688:
C 0A8	2691:	2693:	2695:	2697:	2700:	2702:	2704:	2706:
C 0B0	2709:	2711:	2713:	2715:	2718:	2720:	2722:	2724:
C 0B8	2727:	2729:	2731:	2733:	2736:	2738:	2740:	2742:
C 0C0	2750:	2751:	2752:	2753:	2754:	2755:	2756:	2757:
C 0C8	2760:	2761:	2762:	2763:	2764:	2765:	2766:	2767:
C 0D0	2770:	2771:	2772:	2773:	2774:	2775:	2776:	2777:
C 0D8	2780:	2781:	2782:	2783:	2784:	2785:	2786:	2787:
C 0E0	2791:	2792:	2793:	2794:	2795:	2796:	2797:	2798:
C 0E8	2800:	2801:	2802:	2803:	2804:	2805:	2806:	2807:
C 0F0	2811:	2812:	2813:	2814:	2815:	2816:	2817:	2818:
C 0F8	2821:	2822:	2823:	2824:	2825:	2826:	2827:	2828:
C 100	2836:	2837:	2838:	2839:	2841:	2842:	2843:	2844:
C 108	2846:	2847:	2848:	2849:	2851:	2852:	2853:	2854:
C 110	2857:	2858:	2859:	2860:	2862:	2863:	2864:	2865:
C 118	2867:	2868:	2869:	2870:	2872:	2873:	2874:	2875:
C 120	2878:	2879:	2880:	2881:	2883:	2884:	2885:	2886:
C 128	2888:	2889:	2890:	2891:	2893:	2894:	2895:	2896:
C 130	2899:	2900:	2901:	2902:	2904:	2905:	2906:	2907:
C 138	2909:	2910:	2911:	2912:	2914:	2915:	2916:	2917:
C 140	2920:	2921:	2922:	2923:	2925:	2926:	2927:	2928:
C 148	2930:	2931:	2932:	2933:	2935:	2936:	2937:	2938:
C 150	2941:	2942:	2943:	2944:	2946:	2947:	2948:	2949:
C 158	2951:	2952:	2953:	2954:	2956:	2957:	2958:	2959:
C 160	2962:	2963:	2964:	2965:	2967:	2968:	2969:	2970:
C 168	2972:	2973:	2974:	2975:	2977:	2978:	2979:	2980:
C 170	2983:	2984:	2985:	2986:	2988:	2989:	2990:	2991:
C 178	2993:	2994:	2995:	2996:	2998:	2999:	3000:	3001:
C 180	3011:	3012:	3013:	3014:	3016:	3017:	3018:	3019:
C 188	3021:	3022:	3023:	3024:	3026:	3027:	3028:	3029:
C 190	3032:	3033:	3034:	3035:	3037:	3038:	3039:	3040:
C 198	3042:	3043:	3044:	3045:	3047:	3048:	3049:	3050:
C 1A0	3053:	3054:	3055:	3056:	3058:	3059:	3060:	3061:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 1A8	3063:	3064:	3065:	3066:	3068:	3069:	3070:	3071:
C 1B0	3074:	3075:	3076:	3077:	3079:	3080:	3081:	3082:
C 1B8	3084:	3085:	3086:	3087:	3089:	3090:	3091:	3092:
C 1C0	3095:	3096:	3097:	3098:	3100:	3101:	3102:	3103:
C 1C8	3105:	3106:	3107:	3108:	3110:	3111:	3112:	3113:
C 1D0	3116:	3117:	3118:	3119:	3121:	3122:	3123:	3124:
C 1D8	3126:	3127:	3128:	3129:	3131:	3132:	3133:	3134:
C 1E0	3137:	3138:	3139:	3140:	3142:	3143:	3144:	3145:
C 1E8	3147:	3148:	3149:	3150:	3152:	3153:	3154:	3155:
C 1F0	3158:	3159:	3160:	3161:	3163:	3164:	3165:	3166:
C 1F8	3168:	3169:	3170:	3171:	3173:	3174:	3175:	3176:

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 000	4716:	4657:	4658:	4660:	4662:	4664:	4666:	4668:
U 008	4670:	4672:	4674:	4676:	4678:	4680:	4682:	4684:
U 010	4686:	4688:	4690:	4692:				
U 018	- 04F Unused							
U 050	4725:	4727:	4729:	4732:	4735:	4739:	4741:	4743:
U 058	4745:	4747:	4749:	4751:	4753:	4755:	4757:	4759:
U 060	4761:	4763:	4765:	4767:	4769:	4771:	4772:	4773:
U 068	4774:	4775:	4776:	4777:	4778:	4779:	4780:	4781:
U 070	4811:	4813:	4815:	4816:	4818:	4819:	4821:	4822:
U 078	4824:	4825:	4827:	4828:	4830:	4831:	4833:	4834:
U 080	4836:	4837:	4839:	4840:	4842:	4843:	4845:	4846:
U 088	4848:	4849:	4851:	4852:	4854:	4855:	4857:	4858:
U 090	4860:	4861:	4863:	4864:	4866:	4867:	4869:	4870:
U 098	4872:	4873:	4875:	4876:	4878:	4879:	4881:	4882:
U 0A0	4884:	4885:	4887:	4888:	4890:	4891:	4893:	4894:
U 0A8	4896:	4897:	4899:	4900:	4902:	4903:	4905:	4906:
U 0B0	4908:	4909:	4911:	4912:	4914:	4915:	4917:	4918:
U 0B8	4920:	4921:	4923:	4924:	4926:	4927:	4929:	4930:
U 0C0	4932:	4933:	4935:	4936:	4938:	4939:	4941:	4942:
U 0C8	4944:	4945:	4947:	4948:	4950:	4951:	4953:	4954:
U 0D0	4956:	4957:	4959:	4960:	4962:	4963:	4965:	4966:
U 0D8	4968:	4969:	4971:	4972:	4974:	4975:	4977:	4978:
U 0E0	4980:	4981:	4983:	4984:	4986:	4987:	4989:	4990:
U 0E8	4992:	4993:	4995:	4996:	4998:	4999:	5001:	5002:
U 0F0	5004:	5005:	5007:	5008:	5010:	5011:	5013:	5014:
U 0F8	5016:	5017:	5019:	5020:	5022:	5023:	5025:	5026:
U 100	5028:	5029:	5031:	5032:	5034:	5035:	5037:	5038:
U 108	5040:	5041:	5043:	5044:	5046:	5047:	5049:	5050:
U 110	5052:	5053:	5055:	5056:	5058:	5059:	5061:	5062:
U 118	5064:	5065:	5067:	5068:	5070:	5071:	5073:	5074:
U 120	5076:	5077:	5079:	5080:	5082:	5083:	5085:	5086:
U 128	5088:	5089:	5091:	5092:	5094:	5095:	5097:	5098:
U 130	5100:	5101:	5103:	5104:	5106:	5107:	5109:	5110:
U 138	5112:	5113:	5115:	5116:	5118:	5119:	5121:	5122:
U 140	5124:	5125:	5127:	5128:	5130:	5131:	5133:	5134:
U 148	5136:	5137:	5139:	5140:	5142:	5143:	5145:	5146:
U 150	5148:	5149:	5151:	5152:	5154:	5155:	5157:	5158:
U 158	5160:	5161:	5163:	5164:	5166:	5167:	5169:	5170:
U 160	5172:	5173:	5182:	5184:	5186:	5187:	5189:	5190:
U 168	5192:	5193:	5195:	5196:	5198:	5199:	5201:	5202:
U 170	5204:	5205:	5207:	5208:	5210:	5211:	5213:	5214:
U 178	5216:	5217:	5219:	5220:	5222:	5223:	5225:	5226:
U 180	5228:	5229:	5231:	5232:	5234:	5235:	5237:	5238:
U 188	5240:	5241:	5243:	5244:	5246:	5247:	5249:	5250:
U 190	5252:	5253:	5255:	5256:	5258:	5259:	5261:	5262:
U 198	5264:	5265:	5267:	5268:	5270:	5271:	5273:	5274:
U 1A0	5276:	5277:	5279:	5280:	5282:	5283:	5285:	5286:
U 1A8	5288:	5289:	5291:	5292:	5294:	5295:	5297:	5298:
U 1B0	5300:	5301:	5303:	5304:	5306:	5307:	5309:	5310:
U 1B8	5312:	5313:	5315:	5316:	5318:	5319:	5321:	5322:
U 1C0	5324:	5325:	5327:	5328:	5330:	5331:	5333:	5334:
U 1C8	5336:	5337:	5339:	5340:	5342:	5343:	5345:	5346:
U 1D0	5348:	5349:	5351:	5352:	5354:	5355:	5357:	5358:

ZZ-ENKCH-1.0 :
: ENKCH.MCR :
:

MICRO2 1M(01) Location / Line Num19-MAY-83
Location / Line Number Index

I 13
Fiche 1 Frame I13 Sequence 164
ENKCH Micro-diagnostic for IDC module Rev 01.00 Page 161

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1D8	5360:	5361:	5363:	5364:	5366:	5367:	5369:	5370:
U 1E0	5372:	5373:	5375:	5376:	5378:	5379:	5381:	5382:
U 1E8	5384:	5385:	5387:	5388:	5390:	5391:	5393:	5394:
U 1F0	5396:	5397:	5399:	5400:	5402:	5403:	5405:	5406:
U 1F8	5408:	5409:	5411:	5412:	5414:	5415:	5417:	5418:
U 200	5420:	5421:	5423:	5424:	5426:	5427:	5429:	5430:
U 208	5432:	5433:	5435:	5436:	5438:	5439:	5441:	5442:
U 210	5444:	5445:	5447:	5448:	5450:	5451:	5453:	5454:
U 218	5456:	5457:	5459:	5460:	5462:	5463:	5465:	5466:
U 220	5468:	5469:	5471:	5472:	5474:	5475:	5477:	5478:
U 228	5480:	5481:	5483:	5484:	5486:	5487:	5489:	5490:
U 230	5492:	5493:	5495:	5496:	5498:	5499:	5501:	5502:
U 238	5504:	5505:	5507:	5508:	5510:	5511:	5513:	5514:
U 240	5516:	5517:	5519:	5520:	5522:	5523:	5525:	5526:
U 248	5528:	5529:	5531:	5532:	5534:	5535:	5537:	5538:
U 250	5540:	5541:	5543:	5544:				
U 258 - 5FF Unused								
U 600	5567:	5568:	5570:	5573:	5574:	5575:	5589:	5590:
U 608	5652:	5654:	5656:	5658:	5659:	5714:	5715:	5717:
U 610	5719:	5721:	5724:	5726:	5727:	5728:	5732:	5733:
U 618	5734:	5738:	5739:	5740:	5742:	5745:	5746:	5801:
U 620	5803:	5804:	5807:	5809:	5813:	5815:	5819:	5825:
U 628	5827:	5828:	5834:	5836:	5838:	5839:	5840:	5846:
U 630	5847:	5848:	5849:	5850:	5851:	5852:	5855:	5857:
U 638	5858:	5862:	5912:	5914:	5916:	5918:	5921:	5922:
U 640	5924:	5925:	5980:	5982:	5983:	5986:	5988:	5992:
U 648	5994:	5998:	6004:	6006:	6007:	6013:	6014:	6015:
U 650	6016:	6017:	6018:	6021:	6071:	6073:	6076:	6078:
U 658	6081:	6082:	6084:	6085:	6133:	6138:	6140:	6142:
U 660	6143:	6144:	6194:	6196:	6199:	6203:	6205:	6206:
U 668	6245:	6246:	6247:	6250:	6251:	6252:	6255:	6256:
U 670	6257:	6260:	6261:	6262:	6265:	6266:	6270:	6271:
U 678	6272:	6311:	6312:	6313:	6317:	6318:	6319:	6323:
U 680	6324:	6325:	6329:	6330:	6334:	6335:	6339:	6342:
U 688	6344:	6391:	6392:	6393:	6394:	6395:	6442:	6443:
U 690	6444:	6445:	6446:	6514:	6516:	6519:	6520:	6522:
U 698	6524:	6525:	6526:	6528:	6532:	6534:	6536:	6540:
U 6A0	6541:	6544:	6545:	6547:	6549:	6550:	6552:	6554:
U 6A8	6559:	6560:	6562:	6564:	6565:	6571:	6572:	6575:
U 6B0	6576:	6577:	6582:	6583:	6585:	6586:	6588:	6590:
U 6B8	6591:	6593:	6595:	6597:	6599:	6601:	6608:	6609:
U 6C0	6611:	6614:	6627:	6628:	6630:	6631:	6632:	6633:
U 6C8	6634:	6635:	6636:	6637:	6638:	6642:	6645:	6646:
U 6D0	6647:	6648:	6649:	6650:	6651:	6652:	6653:	6654:
U 6D8	6655:	6656:	6657:	6658:	6659:	6660:	6663:	6666:
U 6E0	6669:	6671:	6672:	6673:	6674:	6677:	6680:	6683:
U 6E8	6685:	6686:	6687:	6689:	6690:	6691:	6693:	6695:
U 6F0	6696:	6697:	6698:	6701:	6702:	6705:	6707:	6708:
U 6F8 - 6FF Unused								
U 700	6781:	6783:	6787:	6789:	6792:	6793:	6794:	6796:
U 708	6797:	6798:	6799:	6805:	6806:	6809:	6810:	6812:
U 710	6813:	6814:	6815:	6816:	6818:	6820:	6823:	6824:
U 718	6827:	6828:	6831:	6832:	6834:	6835:	6836:	6837:

ZZ-ENKCH-1.0
: ENKCH.MCR
:

MICRO2 1M(01)
Location / Line Number Index

J 13

19-MAY-83 13:28:52

Fiche 1 Frame J13
ENKCH Micro-diagnostic for IDC module

Sequence 165
Rev 01.00

Page 162

:Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 720	6838:	6840:	6842:	6845:	6846:	6849:	6850:	6851:
U 728	6853:	6856:	6858:	6859:	6861:	6862:	6863:	6864:
U 730	6865:	6866:	6868:	6872:	6873:	6874:	6875:	6877:
U 738	6885:	6886:	6888:	6890:				

ZZ-ENKCH-1.0 :
: ENKCH.MCR :
:

MICRO2 1M(01) C-code Microword Summary K 13
19-MAY-83 13:28:52 ENKCH Micro-diagnostic for IDC module Rev 01.00 Sequence 166 Page 163

ENKCH Words not
in bounds
512

Used 512
Remaining

Total microwords used in memory C: 512
Total microwords remaining in memory C: 0
Highest address used in memory C: 180 (hex)

Words not
in bounds
ENKCH 844

Used 844
Remaining

Total microwords used in memory U: 844
Total microwords remaining in memory U: 0
Highest address used in memory U: 73B (hex)

; The file used in this assembly is:
ENKCH.MIC

Pass 1 warnings:	0	Pass 2 warnings:	0
Pass 1 errors:	0	Pass 2 errors:	0